НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____Сергій СТІРЕНКО

«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Система підготовки абітурієнтів до вступу»

Виконав:
студент 4 курсу, групи ІП-62
Зварич Євген Олександрович _____

Керівник:
Доцент, к.т.н Габінет Артем Вікторович _____

Консультант з нормо контролю:
Професор, д.т.н Сімоненко Валерій Павлович _____

Рецензент: _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____2020 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Зваричу Євгену Олександровичу

1. Тема проєкту «Система підготовки абітурієнтів до вступу», керівник проєкту доцент, к.т.н. Габінет Артем Вікторович, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки: загальний огляд проблеми, огляд наявних програмних продуктів, огляд обраних для розробки технологій, опис програмної реалізації проекту.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): схема алгоритму роботи програми, схема зв'язків між класами, схема прецедентів.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	06.02.2020	
2	Вивчення та аналіз завдання	15.04.2020	
3	Огляд існуючих програмних рішень	20.04.2020	
4	Огляд можливих технологій	23.04.2020	
4	Розробка структури бази даних	27.04.2020	
5	Розробка серверної частини застосунку	12.05.2020	
6	Розробка клієнтської частини застосунку	20.05.2020	
7	Оформлення пояснювальної записки	25.05.2020	

Студент Євген ЗВАРИЧ

Керівник Артем ГАБІНЕТ

АНОТАЦІЯ

В бакалаврській дипломній роботі реалізовано систему для підготовки абітурієнтів до вступу у вищий навчальний заклад, призначений для школярів, які планують складати вступні іспити або просто покращити свої знання в певних галузях.

Дана система дозволяє користувачам переглядати навчальні матеріали з різних навчальних дисциплін, перевіряти свої знання шляхом проходження тестів, взаємодіяти з іншими користувачами шляхом участі у змаганнях.

Розроблений програмний продукт складається з моделі бази даних, для розробки якої було використано систему керування базами даних MySQL, серверної частини, яка була розроблена з використанням фреймворку Spring, та клієнтської частини, виконаної на фреймворку Angular.

ANNOTATION

In bachelor’s degree thesis was developed system for preparing applicants for university entrance exams, designed for schoolchildren, who are planning to pass entrance exams or just to improve their knowledge in some areas.

This system allows users to view training materials in various subjects, test their knowledge by passing tests, interact with other users by taking part in competitions.

Created programming product comprises database model, made with usage of database management system MySQL, server side, developed using framework Spring, and client side of application, based on framework Angular.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ.467100.001 ВП	Відомість проєкту	1	
2	А4	ІАЛЦ.467100.002 ТЗ	Технічне завдання	4	
3	А4	ІАЛЦ.467100.003 ПЗ	Пояснювальна записка	64	
4	А3	ІАЛЦ.467100.004 Д1	Схема алгоритму роботи програми	1	
5	А3	ІАЛЦ.467100.005 Д2	Схема зв’язків між класами	1	
6	А3	ІАЛЦ.467100.006 Д3	Схема прецедентів	1	
7	А4	ІАЛЦ.467100.007 Д4	Текст програми	10	

					ІАЛЦ.467100.001 ВП		
Зм.	*Арк.*	*ПІБ*	*Підп.*	*Дата*			
Розроб.		Зварич Є.О			Відомість дипломного проєкту	Лист	Листів
Керівн.		Габінет А.В.				1	1
Консульт.						КПІ ім.Ігоря Сікорського кафедра ОТ, гр. ІП-62	
Н/контр.		Сімоненко В.П.					
Зав. Каф.							

Технічне завдання до *дипломної* роботи

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Найменування: «Система підготовки абітурієнтів до вступу». Галузь застосування – школи та підготовчі курси для вступу до вищих навчальних закладів.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр програмної інженерії», затверджене кафедрою обчислювальної техніки Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення програмного забезпечення для майбутніх вступників, викладачів, які хочуть поширювати навчальні матеріали та охочих поглибити свої знання зі шкільних навчальних дисциплін.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є публікації в мережі Інтернет, науково-технічна література, документація програмних засобів, що використовуються при розробці.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту

Система має містити наступний функціонал:

- реєстрація та авторизація користувачів
- можливість переглядати навчальні матеріали
- можливість проходити тести з навчальних дисциплін та їх підтем

- можливість залишати коментарі під матеріалами та пости на власній сторінці
- можливість додавати в друзі інших користувачів
- можливість створювати та брати участь у змаганнях з іншими користувачами
- можливість додавати нові навчальні матеріали та тестові запитання (для викладачів)
- можливість змінювати роль користувача та видаляти його обліковий запис (для адміністратора)

5.2. Вимоги до програмного забезпечення

Мінімальний набір програмного забезпечення для запуску системи:

- операційна система Windows версії 7 або вище, будь-який дистрибутив Linux (Ubuntu, Fedora, Red Hat) або MacOS
- Java Runtime Environment версії 8 або вище для запуску та виконання Java-застосунків
- будь-який інтернет-браузер з увімкненою можливістю виконання Javascript-коду (Google Chrome, Mozilla Firefox, Microsoft Edge)

5.3. Вимоги до апаратної частини

Мінімальна необхідна кількість операційної пам’яті – 2 Гб, мінімальна кількість вільного місця на диску – 2 Гб, необхідне підключення до мережі Інтернет.

6. ЕТАПИ РОЗРОБКИ

	Дата
Затвердження теми роботи	06.02.2020
Вивчення та аналіз завдання	15.04.2020
Огляд існуючих програмних рішень	20.04.2020
Огляд можливих технологій	23.04.2020
Розробка структури бази даних	27.04.2020
Розробка серверної частини застосунку	12.05.2020
Розробка клієнтської частини застосунку	20.05.2020
Оформлення пояснювальної записки	25.05.2020

Пояснювальна записка
до дипломного проекту

на тему: СИСТЕМА ПІДГОТОВКИ АБІТУРІЄНТІВ ДО ВСТУПУ

Київ – 2020 року

ЗМІСТ

СПИСОК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ЗНО – Зовнішнє незалежне оцінювання – іспити для вступу до вишів в Україні [1].

Spring – фреймворк з відкритим вихідним кодом для розробки додатків різної складності з використанням мови Java.

СКБД – система керування базами даних.

Angular – фреймворк з відкритим вихідним кодом для розробки користувацького інтерфейсу веб-додатків з використанням мови програмування TypeScript.

MVC (Model-View-Controller) – шаблон проектування програмних застосунків, який полягає у поділі системи на три взаємопов’язаних частини, кожну з яких можна модифікувати незалежно від інших: модель, представлення та контролер [2].

Впровадження залежностей – шаблон проектування програмних застосунків, який дозволяє надавати компоненту програми залежності від інших компонентів, найчастіше виконується фреймворком, який робить це автоматично при запуску або оновленні програмного коду. Залежність – зв'язок між програмними компонентами, один з яких можна використати при виконанні коду іншого.

ВСТУП

В наш час склалася суспільна думка про те, що здобуття вищої освіти для будь-якої людини є обов'язковим елементом, необхідним для кар'єрного зростання, самореалізації та інтелектуального розвитку. Також не можна заперечувати, що саме в закладі вищої освіти формується набір базових навичок і вмінь для представників переважної більшості висококваліфікованих професій, таких як інженери, лікарі, юристи, фінансисти. Сукупність цих і не тільки факторів призводить до того, що, згідно з заявою першого заступника міністра освіти України, 79% українців отримують вищу освіту [3].

Проте перед тим, як абітурієнти побачать свої прізвища у списку зарахованих до бажаного навчального закладу, їм необхідно скласти вступні іспити, які відбуваються у форматі зовнішнього незалежного тестування. Кожен випускник обирає власний формат підготовки, дехто користується послугами репетиторів, хтось використовує лише інформацію, отриману від викладачів у школі, але все популярнішим стає онлайн-навчання, тобто отримання інформації з навчальних відео та статей з мережі Інтернет. Але ця модель отримання інформації має також і свої недоліки, пов'язані з недостатністю комунікації між викладачем та учнем та можливим зникненням інтересу в учня за відсутності інших мотивуючих факторів, окрім власне майбутнього вступу до ВНЗ.

Вказані проблеми можуть бути вирішені за допомогою додавання до процесу навчання інтерактивного елементу та формування платформи для комунікації між вчителями та учнями та між власне учнями. Завданням даної дипломної роботи буде саме створення системи, що дозволить об'єднати переваги вже існуючих систем для онлайн-навчання з виправленням їх недоліків.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ

1.1 Загальні положення

Починаючи з 2004 року, вступ до вищих навчальних закладів здійснюється на підставі результатів зовнішнього незалежного оцінювання (ЗНО) – комплексу заходів (переважно – тестування), спрямованих на визначення рівня знань випускників шкіл [1]. Впровадження ЗНО допомогло покращити навчальні показники учнів та студентів закладів середньої та вищої освіти відповідно, стало найвагомішим фактором в боротьбі з корупційним чинником при вступі (раніше абітурієнтам необхідно було складати вступні іспити безпосередньо в навчальних закладах, в які вони планували вступити для здобуття вищої освіти, що стало джерелом корупції).

Кожен заклад вищої освіти формує список необхідних ЗНО для вступу на відповідну спеціальність і мінімальний прохідний бал. Станом на 2020 рік проводяться тестування з таких дисциплін: українська мова і література, математика, історія України, фізика, хімія, біологія, географія, іноземна мова (на вибір англійська, французька, німецька або іспанська).

При підготовці до ЗНО учні використовують декілька найочевидніших можливостей для покращення їх знань. Такими є самоосвіта в рамках шкільної програми, використання репетиторів та навчання онлайн з використанням спеціалізованих відео або курсів. Згідно з даними опитування 2016 року, 38% випускників готувалися до іспитів самостійно під час занять у школі, 17% проходили безкоштовні онлайн/офлайн курси і 24% не користувалися жодною сторонньою допомогою [4]. З огляду на стрімку діджиталізацію суспільства, можна висунути припущення, що кількість учнів, що використовують онлайн-платформи, зростає і ще не досягла свого піку.

1.2 Загальний аналіз проблеми онлайн-навчання

Сучасна система освіти разом з усім суспільством переживає якісні зміни, що відбуваються внаслідок розвитку інформаційних технологій і їх впливу на всі сфери нашого життя. Внаслідок цього поступово виникають зміни на рівнях загальної середньої та вищої освіти і загалом у засобах отримання необхідної інформації. Однією з найоптимальніших форм сучасного навчання починає виявлятися дистанційна форма навчання, яка багатьма дослідниками вже названа «системою 21 століття». Саме вона надає можливість створення систем безперервного самонавчання, що є найпрогресивнішим та найбільш пристосованим по потреб сучасного суспільства рішенням у сфері освіти.

Дистанційна освіта має ряд чітко виражених переваг у порівняння з класичною:

– економічна ефективність (покращене співвідношення досягнутого результату до витрачених економічних та інших ресурсів, витрачених на організацію навчання);

– можливість коректування навчального матеріалу для його кращого засвоєння за певний період часу;

– гнучкість (можливість визначити зручне місце, час та темп навчання);

– масовість (кількість учнів не обмежена), модифікована роль викладача в навчальному процесі (коректування змісту дисципліни, консультаційна роль, загальне координування процесу);

– соціальність (порівняно рівні можливості для здобуття знань незалежно від місця проживання та матеріального становища).

Одним з найбільш інноваційних напрямків у галузі дистанційного навчання є онлайн-навчання в цілому і онлайн-курси зокрема. Онлайн-курс – це інтернет-курс з відкритим доступом та інтерактивною участю, що дозволяє будь-кому засвоїти ту чи іншу дисципліну і скласти іспит у режимі онлайн. Додатково до звичних матеріалів курсу, як-то лекції та домашні завдання, учні отримують

доступ до інтерактивного форуму користувачів і стають частиною спільноти студентів, вчителів та асистентів(TAS).

На сьогоднішній день усі існуючі онлайн-курси можна розділити на три великі групи:

– масові відкриті онлайн-курси, що використовують за основу конективістський підхід. В таких курсах ціль навчання визначається самою особою, що навчається. Експерти вважають, що найкраще даний тип курсів використовувати людям, що є мотивованими на самонавчання і можуть самостійно обирати необхідні ресурси. Найефективнішою сферою застосування даної категорії курсів є підвищення власної кваліфікації;

– масові відкриті онлайн-курси, що базуються на певних завданнях. Основою функціонування даного типу курсів є можливість слухача обирати та виконувати певний набір завдань самостійно або в кооперації з іншими слухачами курсу;

– масові відкриті онлайн-курси, що мають чіткі графіки. Такий тип курсів найчастіше використовується у великих міжнародних університетах, розробкою таких курсів найчастіше займаються викладачі, наукові співробітники та визнані експерти в своїх галузях. Дані курси характеризуються чітким навчальним графіком, планом, розкладом дедлайнами для слухачів.

1.3 Опис існуючих рішень

Для порівняння можна розділити існуючі рішення на дві категорії: ті, що надають послуги з онлайн-навчання загалом, та ті, що дозволяють підготуватися власне до ЗНО та мають більш вузьку спеціалізацію. Далі будуть розглянуті найвідоміші програмні продукти для кожної з цих категорій.

1.3.1 Приклади платформ для онлайн-навчання

Усі найвідоміші платформи для онлайн-навчання мають схожий набір базового функціоналу, наприклад авторизація користувача, можливість

реєструватися на курс, для викладача – можливість створювати власні курси, тести, проте вони відрізняються за моделлю монетизації (безкоштовні - платні), наявністю локалізацій (підтримують лише англійську, російську мову чи є багатомовними), цільовою аудиторією (для студентів, для корпоративного сектору).

а) Moodle (https://moodle.org/) – безкоштовна відкрита (Open Source) платформа для організації дистанційного навчання, створена у 2001 році австралійцем Мартіном Дугіамасом. Система написана на мові PHP з використанням реляційних баз даних (MySQL, PostgreSQL). Головною перевагою є сильна спільнота розробників платформи, яка постійно збільшується і саме вона здійснює підтримку вже існуючих версій та розвиток нових, створює нові модулі (плагіни). Наразі Moodle перекладено на більш, ніж 100 мов світу, зокрема й українською, і підтримує більше 1500 плагінів. Існує безкоштовна версія, якої цілком достатньо для повноцінної організації онлайн-курсу, і платна, в якій додаються можливості видачі сертифікату, зміни дизайну, підтримки сторонніх плагінів.

Особливості Moodle: налаштування платформи через плагіни, які можна скачати з Інтернету або створити самому, інтеграція з іншими сервісами та платформами, наприклад, системою управління контентом Wordpress та системою проведення відеоконференцій Zoom. Переваги для студентів: широкі можливості для комунікацій та групової роботи(форум, чат, wiki), можливість

отримувати нагадування про події, здійснюється повне відслідковування прогресу студента (результати проходження курсу, тесту тощо)

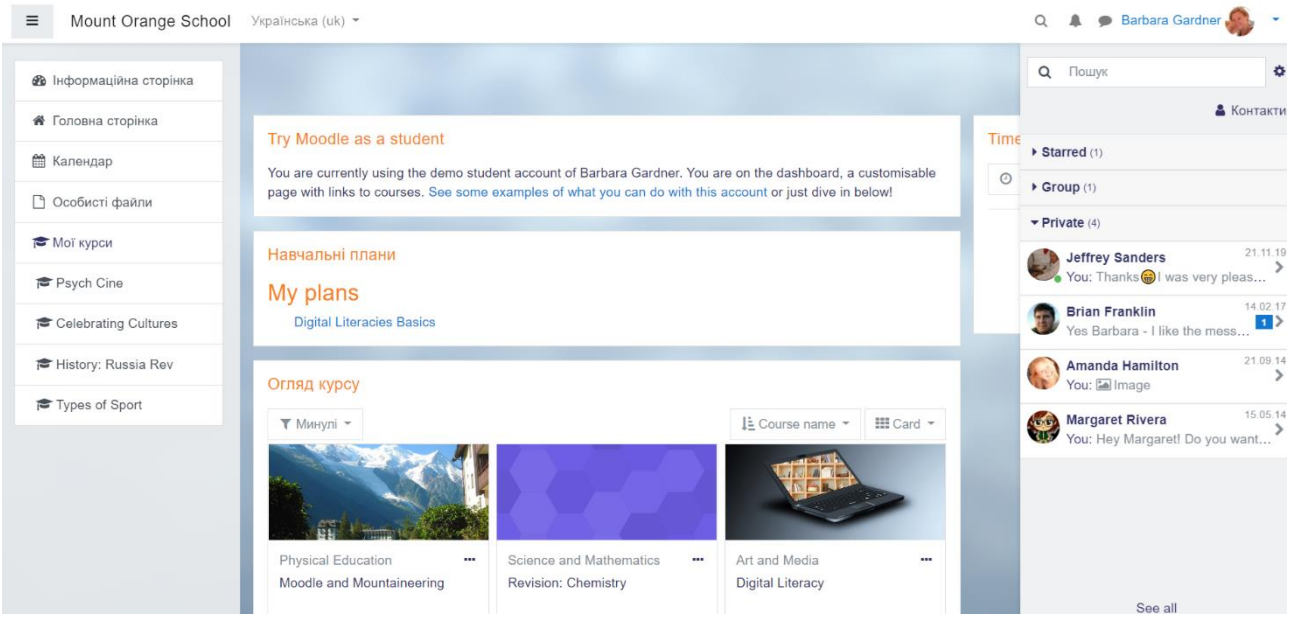


Рис.1.1 Сторінка студента на ресурсі Moodle

Переваги для викладачів: наявність великої кількості інструментів для створення авторських курсів, можливість розміщення навчальних матеріалів у різних форматах(.doc, .html, .pdf, відео- та аудіофайли), автоматизований процес перевірки знань студентів(автостворення тестів різних типів).

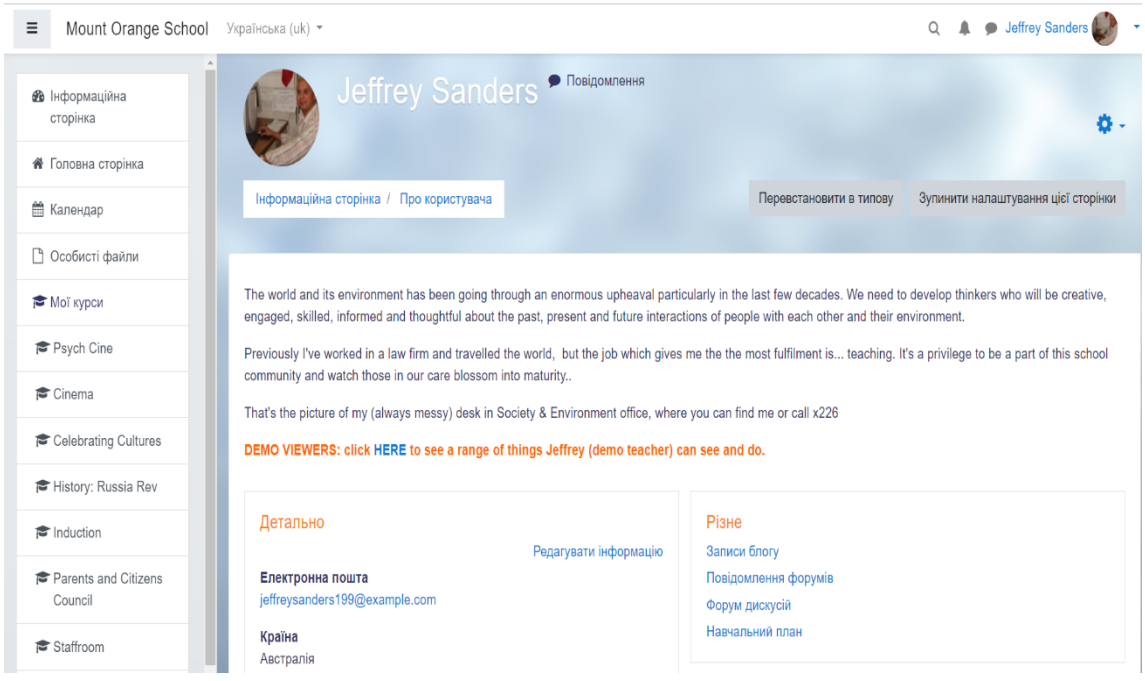


Рис. 1.2. Особиста сторінка користувача на ресурсі Moodle

Незважаючи на численні переваги даного рішення, можна відзначити також суттєві недоліки. Найважливіші з них – надмірна складність системи для новачка, довгий процес налагоджування і доопрацювання системи, необхідність купувати і підтримувати власний хостинг для системи. Внаслідок цього можна зробити висновок, що Moodle як платформа більше підходить для організації онлайн-навчання у локальному середовищі (закладі загальної середньої або вищої освіти) з виділеними потужностями для адміністрування та наповнення сайту, з обмеженою кількістю студентів та 1-2 викладачами, що постійно повинні перебувати з ними у контакті, а не для глобальної платформи, якою можуть користуватися тисячі студентів і викладачів, які постійно обмінюються знаннями і потребують простого та зрозумілого каналу спілкування.

b) ATutor (https://atutor.ca/) – система керування навчанням, створена у 2001 році професорами з Дослідницького центру адаптивних технологій Університету Торонто, також розвивається як open source продукт. Має схожий базовий функціонал, як і Moodle, але не настільки розвинену спільноту, внаслідок чого не має настільки великих можливостей. Присутні всі базові функції для проведення онлайн-курсів, тестування, спілкування між викладачами та студентами.

Рис. 1.3. Скріншот з ресурсу ATutor

1.3.2 Приклади платформ для підготовки до ЗНО

Оскільки процес вступу до закладу вищої освіти відчутно відрізняється в різних країнах, а, відповідно, відрізняється і інформація, необхідна для засвоєння, то немає сенсу порівнювати платформи, створені в інших країнах з платформами, орієнтованими на українського користувача, тому зосередимося саме на продуктах, виготовлених в Україні

а) ILearn (https://ilearn.org.ua/) – продукт, створений громадською спілкою «Освіторія», яка займається розвитком освіти в Україні, в 2013 році. З часу його створення його інформаційними ресурсами скористалися понад 60000 абітурієнтів [7]. Даний ресурс позиціонується як безкоштовна гейміфікована платформа з навчальними вебінарами, тестами, онлайн-курсами і має на меті надання усім абітурієнтам України рівних можливостей для вступу до закладів вищої освіти, незалежно від місця проживання та соціального статусу. Оскільки даний ресурс запускався як освітній соціальний проект, він є абсолютно безкоштовним. На даний момент на платформі наявні матеріали для підготовки з української мови і літератури, математики, історії України, біології, англійської мови, а також матеріали на профорієнтаційну тематику.

Основною перевагою даного продукту є повна гейміфікація процесу навчання (додавання ігрових елементів), що робить його інтерактивним і дозволяє захопити увагу учня. Під час реєстрації користувач створює власного віртуального героя, після чого опрацьовує навчальні матеріали начебто від його імені. В платформі присутня віртуальна валюта, яку можна заробити лише шляхом проходження тестів та відвідування вебінарів, яку можна витрачати на покращення зовнішнього вигляду персонажа. Також існує можливість відслідковувати власний прогрес та контактувати з іншими користувачами шляхом додавання їх у друзі.

Серед недоліків цієї платформи слід відзначити неможливість контактувати з викладачами курсів, залишати відгуки про матеріали і загалом недостатню кількість можливостей комунікувати з іншими користувачами,

обговорювати отриману інформацію, ділитися корисними ресурсами. Також важливою є повна статичність курсів, тобто відсутність можливості редагувати матеріали курсів, наприклад, додавати нові відео та питання до тестів та змінювати існуючі, внаслідок чого виникає повне покладання на відсутність помилок адміністраторів ресурсу.

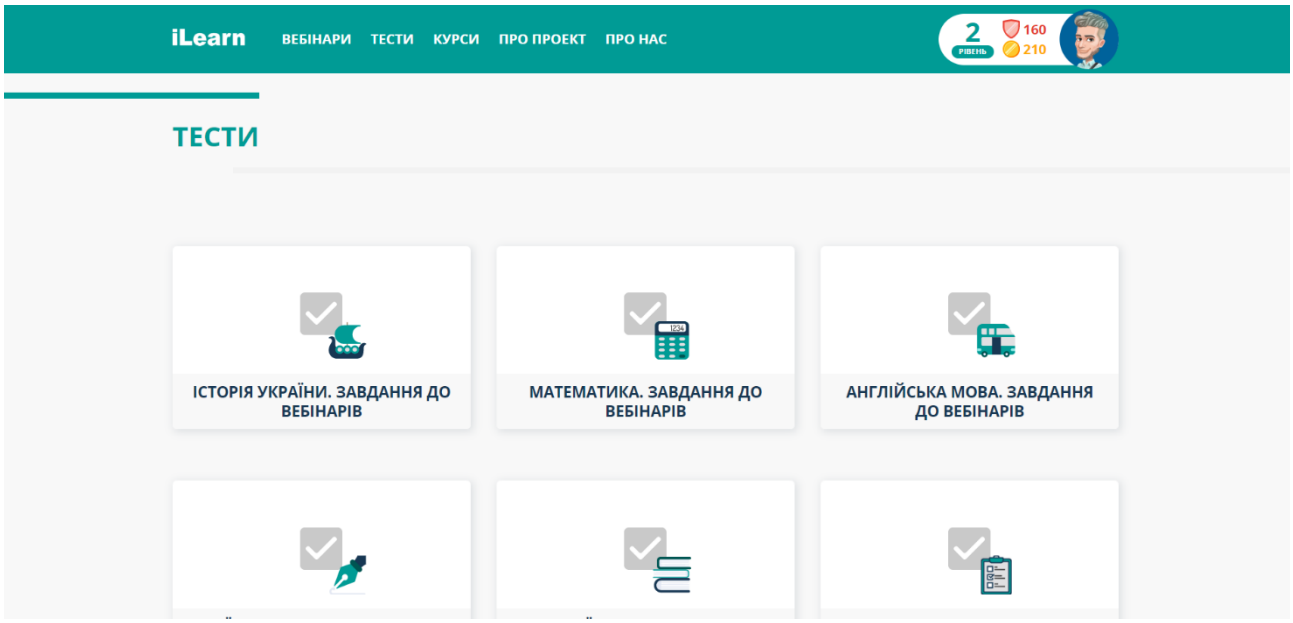


Рис. 1.4. Скріншот сторінки ресурсу https://ilearn.org.ua/

b) Zno.osvita.ua (https://zno.osvita.ua/) – ресурс, що є частиною найпопулярнішого сайту про стан справ освіти в Україні – osvita.ua. На даному сайті можна знайти всю необхідну інформацію про вступну кампанію, терміни проведення вступних іспитів, рейтинги вишів, сформовані за різними параметрами, таблиці для розрахунку рейтингових балів та багато іншого. Хоча даний ресурс не є платформою для онлайн-навчання в класичному розумінні, його необхідно відзначити через те, що лише тут зосереджено всю важливу офіційну інформацію стосовно ЗНО, яку складно знайти на інших ресурсах.

Серед переваг даного продукту можна відзначити наявність бази усіх запитань, що були використані під час проведення основних сесій ЗНО, додаткових сесій і пробних ЗНО минулих років та можливість проходити реальні тести і отримати оцінку своїх результатів у порівнянні з абітурієнтами попередніх років.

Недоліків при розгляді даної платформи з позиції онлайн-навчання можна помітити набагато більше, ніж переваг. По-перше, відзначимо повну відсутність ресурсів для підготовки до іспитів, наявність комунікації між користувачами лише шляхом спілкування у коментарях під новинами, відсутність будь-якого інтерактивного елементу при навчанні. Тобто, можемо зробити висновок, що даний ресурс чудово підходить у якості допоміжного, але аж ніяк не основного при побудові програми підготовки до тестування.

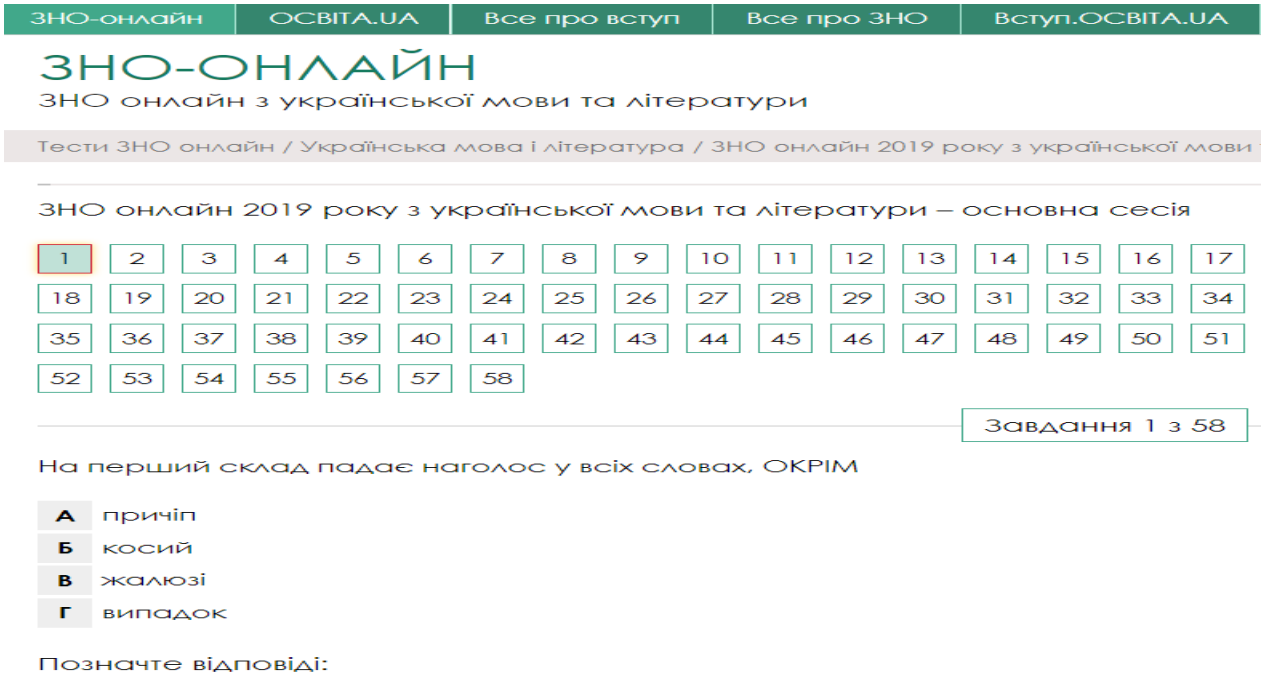


Рис. 1.5. Скріншот сторінки ресурсу https://zno.osvita.ua/

1.3.3 Освітня платформа Prometheus

Обидві вимоги (платформа для онлайн-навчання та можливість підготуватися до ЗНО) об’єднує в собі українська платформа Prometheus (https://prometheus.org.ua/) – неприбутковий волонтерський проект для проведення масових онлайн-курсів. Даний ресурс було засновано наприкінці 2014 року аспірантом КНУ імені Тараса Шевченка Іваном Примаченком та викладачем КПІ імені Ігоря Сікорського Олексієм Молчановським, за перші півроку на платформі було зареєстровано 70000 користувачів. Наразі за допомогою курсів даного ресурсу можна отримати базові навички у сферах

громадянської освіти, програмування, аналізу даних, підприємництва та інших. Курси викладаються переважно професорами найкращих університетів України та практикуючими спеціалістами у різних галузях знань. Також на даній платформі присутній цикл курсів з підготовки до ЗНО, який включає в себе курси з української мови та літератури, історії України, математики та англійської мови.

Дана платформа має ряд беззаперечних переваг, серед яких вдала комбінація відеолекцій, тестувань для перевірки знань та практичних завдань, наявність платформи для комунікацій між викладачами (засновниками курсу) та студентами (користувачами), можливість отримати конспект лекцій, завантажити презентацію за матеріалами відеолекції, залишати коментарі під відео.

Серед недоліків можна відзначити те, що користувачі можуть комунікувати одне з одним лише в рамках одного курсу, відсутність можливості додавати в друзі, відсутність будь-якого інтерактивного елементу і недостатню гнучкість курсів (неможливість внесення додаткової інформації в курс для викладачів, що не є його засновниками).

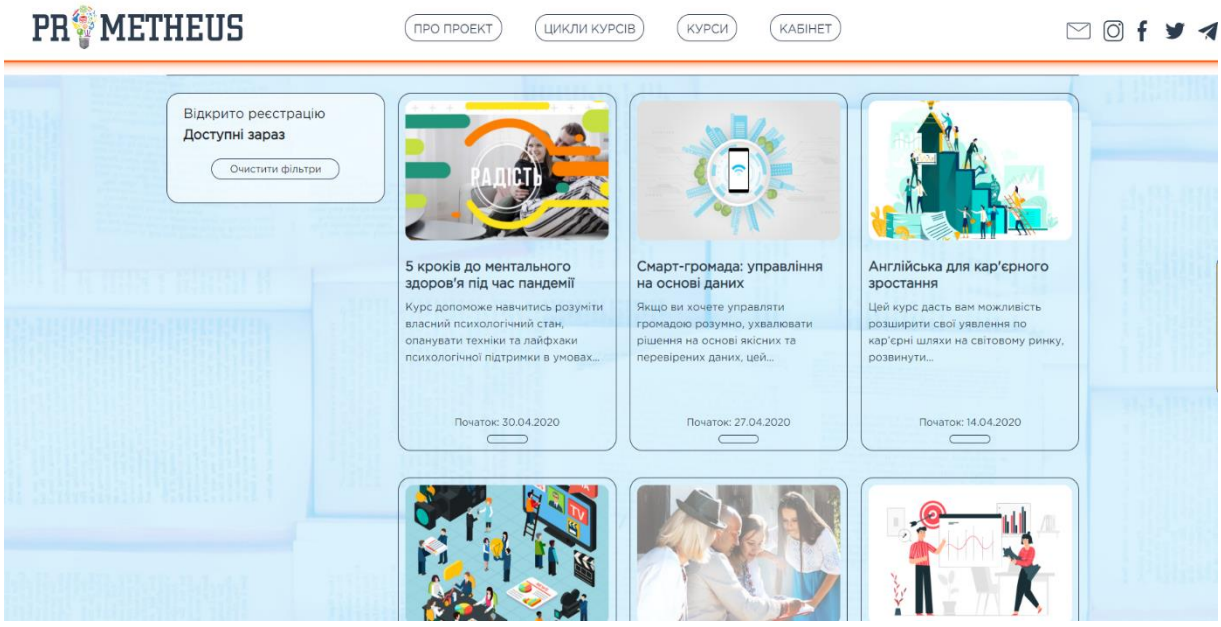


Рис. 1.6. Скріншот сторінки ресурсу https://prometheus.org.ua/

1.4 Аналіз вимог до продукту

На підставі проведеного аналізу наявних рішень проблеми, що досліджується, можемо визначити основні вимоги до продукту.

Основними ролями користувачів в межах даного проекту є студенти (слухачі, учні, здобувачі знань) і викладачі.

Користувачі поділяються на авторизованих та неавторизованих. Неавторизовані – усі відвідувачі до реєстрації та логіну в систему. Неавторизовані користувачі мають можливість бачити контент лише головної сторінки ресурсу, авторизовані – отримують доступ та набір можливостей для роботи з ресурсом відповідно до ролі.

Авторизовані користувачі поділяються на три ролі: студенти, викладачі та адміністратори. Студенти мають можливість:

- переглядати контент всіх навчальних матеріалів;
- проходити тести з певної дисципліни або навчальної теми;
- додавати в друзі інших користувачів;
- залишати пости на власній сторінці;
- залишати коментарі під навчальними відео;
- переглядати таблиці найкращих результатів проходження навчальних тестів;
- створювати і брати участь у змаганнях з іншими користувачами.

Викладачі мають можливість здійснювати усі дії, які дозволені студентам, а також можуть поповнювати базу навчальних матеріалів (додавати та видаляти навчальні відео, додавати та видаляти тестові запитання різного формату) і додавати нові теми в існуючих навчальних дисциплінах.

Адміністратор має всі можливості, які надані попереднім двом ролям, а також може додавати нові навчальні дисципліни і виконувати різні дії з акаунтами користувачів (зміна ролі, видалення користувача за певні дії).

Навчальні відео – основне джерело інформації для студентів-користувачів системи, додаються викладачами шляхом збереження посилання на зовнішній

ресурс, відео з якого буде показано у вбудованому відео-програвачі всередині даної платформи.

Тестові запитання – основне джерело перевірки поточних знань студентів, існує можливість створити запитання з однією правильною відповіддю, декількома правильними відповідями та з відкритою відповіддю.

Висновки до розділу 1

Виходячи з усього вищенаведеного, можна дійти висновку, що предметна область, що розглядається в рамках даної роботи, на сьогодні є досить затребуваною серед широкого кола користувачів, з огляду на їх зацікавлення в якісній самоосвіті та успішному складанні ЗНО зокрема.

В умовах стрімкої діджиталізації в Україні перехід до здобуття онлайн-освіти є особливо актуальним і сьогодні вже існують програмні рішення, які в тій чи іншій мірі вирішують поставлену проблему. Проте на даний момент жоден з наведених продуктів не позбавлений недоліків, які заважають стати монопольним лідером у сфері онлайн-підготовки абітурієнтів до вступних іспитів. В результаті виконання даної дипломної роботи буде створено продукт, в якому об’єднано переваги усіх наведених програмних рішень та враховано їх найсуттєвіші недоліки.

РОЗДІЛ 2
ОГЛЯД ОБРАНИХ ТЕХНОЛОГІЙ

Опис технологій, які були використані для даного проекту, доцільно розділити на дві частини: серверна частина, яка відповідає за бізнес-логіку застосунку та роботу з базою даних, та клієнтську, на якій відбувається відображення даних, що отримані з серверної частини.

2.1 Серверна частина

2.1.1 Загальні поняття

2.1.1.1 Сервер

Сервер – комп’ютер, виділений для виконання певної сервісної задачі або для встановлення певного сервісного програмного забезпечення. Найчастіше під назвою «сервер» мають на увазі веб-сервер – машину, що приймає HTTP-запити від клієнтів і повертає їм HTTP-відповіді, які можуть містити HTML-сторінку, зображення, файли або іншу інформацію. Для того, щоб програмний ресурс міг використовуватися у мережі Інтернет, його програмний код необхідно завантажити на віддалений хостинг. Для розробки і налагодження програми на локальній машині розробника використовується так званий локальний сервер.

В якості клієнтів веб-сервера можуть слугувати різні програми і пристрої. Найчастіше це веб-браузер, що працює на ПК або мобільному пристрої. Також це можуть бути різні програми, що звертаються до веб-серверів для отримання певних оновлень або іншої інформації, необхідної для продовження роботи (наприклад, антивіруси так оновлюють свої бази даних) або мобільні телефони, що отримують доступ до веб-серверу через протокол WAP.

На сьогоднішній день існує велика кількість веб-серверів, які можна використовувати для розгортання веб-ресурсів, деякі з них:

- Apache HTTP Server;
- Apache Tomcat;

- CERN httpd – історично перший створений веб-сервер;
- Jetty;
- Nginx.

2.1.1.2 HTTP

HTTP (англ. Hypertext Transfer Protocol – «протокол передачі гіпертексту») – протокол прикладного рівня передачі даних (7 рівень моделі OSI) спочатку у вигляді гіпертекстових документів у вигляді HTML-файлів, а надалі і для передачі будь-яких довільних даних. Основою HTTP є технологія «клієнт-сервер», тобто припускається існування користувачів, які ініціюють з'єднання и надсилають запит, та постачальників (серверів), які очікують на з'єднання, виконують необхідні дії та повертають повідомлення з результатом.

Основним об'єктом в HTTP є ресурс, на який вказує URI (Uniform Resource Identifier) в запиті клієнта. Особливістю даного протоколу є можливість вказати в запиті та відповіді спосіб представлення одного і того самого ресурсу за різними параметрами, як-то форматом, кодуванням.

Кожне HTTP-повідомлення складається з трьох частин: стартовий рядок (визначає тип повідомлення), заголовки (характеризують тіло повідомлення, параметри передачі тощо), тіло повідомлення (дані, обов'язково відділяються від заголовків пустим рядком).

Стартовий рядок відрізняється для запиту та відповіді. Для запиту він виглядає як «Метод URI HTTP/Версія». Методи, які використовуються HTTP, будуть розглянуті нижче. Для відповіді «HTTP/Версія КодСтану Пояснення». Код стану – три цифри, які визначають наступну поведінку клієнта, також будуть розглянуті нижче. Пояснення – текстове пояснення до коду стану для користувача, є необов'язковим.

2.1.1.3 HTTP методи

HTTP метод – послідовність символів, яка вказує на основну операцію над ресурсом, до якого отримується доступ.

Метод OPTIONS – використовується для визначення можливостей веб-серверу або параметрів з’єднання для конкретного ресурсу. У відповідь серверу потрібно включити заголовок Allow зі списком методів, що підтримуються.

Метод GET – використовується для запиту на читання даних з ресурсу або для початку певного процесу. Клієнт може передавати параметри запиту в URI після символу “?”. Приклад: google.com/search?q=сервер.

Метод HEAD – аналогічний методу GET, за винятком того, що у відповіді відсутнє тіло. HEAD-запит зазвичай використовується для отримання метаданих або перевірки наявності ресурсу.

Метод POST – використовується для передачі даних користувача ресурсу. Дані, які необхідно передати на сервер, включаються до тіла запиту.

Метод PUT – також використовується для передачі певних даних на сервер, різниця з методом POST – у розумінні призначення URI ресурсів. Метод POST означає, що за вказаним URI буде виконана обробка тіла запиту, при використанні PUT припускається, що вміст тіла запиту відповідає ресурсу за вказаним URI.

Метод PATCH – є аналогічним PUT, але PUT застосовується для цілого ресурсу, а PATCH – лише для його частини.

Метод DELETE – видаляє вказаний ресурс.

2.1.1.4 Коди стану HTTP

Код стану є частиною відповіді серверу. Він є числом, яке складається з трьох цифр. Перша цифра вказує на клас стану. За кодом відповіді зазвичай слідує фраза, що пояснює причину саме такої відповіді. Приклад: 404 Not Found.

Набір кодів є стандартизованим, нові коди можуть бути додані лише після узгодження з IETF (інженерна рада Інтернету). Наразі виділено п’ять класів кодів:

- 1XX – інформаційний клас – інформує про процес передачі даних, повідомлення від сервера містять лише стартовий рядок відповіді та, за необхідності, декілька специфічних полів заголовку;

- 2XX – успіх – інформує про успішне опрацювання запиту клієнта;
- 3XX – перенаправлення – інформує про те, що для успішного виконання операції необхідно виконати запит на інший URI;
- 4XX – помилка клієнта – вказує на помилку з боку клієнта, для усіх методів, окрім HEAD, сервер повинен повернути в тілі пояснення для користувача;
- 5XX – помилка сервера – інформує про невдало виконану операцію з вини сервера, для усіх методів, окрім HEAD, сервер повинен повернути в тілі пояснення для користувача.

2.1.2 Обґрунтування обраного рішення

На даний момент існує велика кількість можливих інструментів для написання серверної логіки веб-додатку. Найпопулярнішими мовами програмування для цього є PHP, Python, Ruby, Java, C#, Javascript. Усі вказані мови мають свої переваги, недоліки і широкий набір бібліотек та фреймворків, проте для написання серверної частини даного проекту буде використано фреймворк Spring на основі мови Java.

Переваги мови Java:

- кросплатформеність, яка дозволяє запускати Java-програму на будь-якому сервері, незалежно від операційної системи, що встановлена на сервері(наприклад, мова C# є орієнтованою на Windows-системи);
- можливість використовувати переваги об’єктно-орієнтованого підходу до розробки програмного забезпечення;
- багатопоточність – можливість використовувати усі надані ресурси сервера, що допоможе при збільшенні навантаження на систему;
- строга типізація – набір базових типів мови допомагає більш точно описувати предметну область і допомагає у підтримці та виправленні проблем у роботі програми.

2.1.3 Spring

Spring Framework – фреймворк для розробки програмних застосунків на мові програмування Java, який на сьогоднішній день є найпопулярнішим застосуванням для написання Java-застосунків різної складності. Перша версія з'явилася у 2002, після чого розпочалося стрімке зростання платформи через її open-source природу, яка дозволяє брати участь у розробці великій кількості програмістів. Оновлення платформи є регулярними, станом на травень 2020 року основною версією є 5.2.6. Назва Spring є загальною для цілого ряду невеликих фреймворків, кожен з яких охоплює невелику частину розробки застосунку. Деякі з них:

- Spring Core – основний модуль, надає базові засоби для створення додатку – управління Spring-компонентами (за Spring-термінологією вони називаються бінами - beans), впровадження залежностей в компоненти.
- Spring MVC – надає підтримку шаблону проектування MVC для розробки веб-орієнтованих додатків, також містить набір функцій для роботи безпосередньо з HTTP-запитами (дозволяє формувати заголовки, тіло і накладати їх на Java-об'єкти).
- Spring AOP – реалізує підтримку аспектно-орієнтованої парадигми програмування, яка дозволяє покращити модульність застосунку шляхом додавання наскрізних процедур, і обійти певні обмеження, що накладаються об'єктно-орієнтованою парадигмою, яка є основною при розробці на мові Java.
- Spring Data – надає набір функцій для роботи з базами даних (реляційними та нереляційними), для динамічного створення запитів до бази, для програмного створення таблиць та зв'язків між ними.
- Spring Security – надає механізми побудови систем авторизації та аутентифікації, а також інші можливості забезпечення безпеки для застосунків, створених на основі Spring Framework, перша версія була створена у 2003 році під назвою «Acegi Security» незалежно від Spring, проект був поглинутий Spring у 2008 і став його офіційним дочірнім проектом.

- Spring Cloud – надає широкий набір можливостей для запуску застосунку у хмарних сервісах, для створення та підтримки проектів, орієнтованих на мікросервісну архітектуру.
- Spring Test – надає підтримку класів для написання юніт-тестів та інтеграційних тестів.

Основними перевагами Spring можна вважати:

- Spring надає каркас додатку і диктує основні правила побудови архітектури, в які необхідно вбудувати свою функціональність.
- Spring підтримує модульну структуру побудови застосунку, з великої кількості Spring-проектів, які були перелічені вище, розробник може обрати лише ті, які необхідні йому для вирішення поставленого завдання і не використовувати інші.
- в застосунку на основі Spring здійснюється оперуваннями об’єктами, які є слабко зв’язаними за рахунок використання принципу впровадження залежностей, що спрощує тестування застосунку та його модифікацію за зміни вимог до продукту.
- детальна документація усього присутнього функціоналу кожного з елементів фреймворку, розміщена розробниками на офіційному сайті.
- Spring, як вже було зазначено, є найпопулярнішим фреймворком для розробки застосунків на Java, що означає, що існує велика спільнота розробників по всьому світу, які мають знання даної платформи і сформували значну базу знань, в якій є можливість знайти відповідь на будь-які запитання, що виникають в процесі розробки.

Раніше значним недоліком Spring вважалася порівняна складність налаштування проекту перед його запуском та складна інтеграція з системами, які не були розроблені спільнотою Spring (існувала ймовірність, що версія сторонньої бібліотеки є несумісною з версією Spring). Існувала необхідність налаштовувати велику кількість конфігураційних файлів, які могли бути створені з використанням XML-файлів або конфігураційних Java-файлів, що

викликало додаткові складності та підвищений поріг входу у дану технологію. Наразі дана проблема є вирішеною через створення у 2014 році нового проекту у фреймворку Spring – Spring Boot. Він надає набір додаткових можливостей, які значно зменшують час, що необхідно витратити на конфігурування застосунку і дозволяє швидше перейти власне до розробки. Переваги Spring Boot:

- автоконфігурування – Spring Boot самостійно конфігірує додаток, опираючись на залежності, які підключені на даний момент. Наприклад, при додаванні залежності MySQL буде автоматично створена конфігурація для роботи з базою даних MySQL, яку, за необхідності, можна перевизначити або доповнити у користувацьких класах.
- вбудований веб-сервер, який Spring Boot надає та налаштовує автоматично і який запускається при запуску застосунку, за замовчуванням це Apache Tomcat, але можна обрати інший, вказавши його у конфігураційному файлі;
- вводиться поняття стартер-пакету – набір бібліотек, які встановлюються за замовчуванням при підключенні пакету, що відповідає за певну функціональність застосунку. Наприклад, при підключенні пакету spring-boot-starter-data-jpa, Spring Boot автоматично підключить набір необхідних залежностей для роботи з SQL базою даних, а також найпопулярнішу реалізацію технології JPA (Java Persistence API) – Hibernate і зконфігурує їх між собою для готового використання.
- автоматично створюється набір URI-ресурсів для моніторингу стану застосунку, при доступі до яких можна отримати інформацію про кількість пам’яті, яка використовується на даний момент, список компонентів, якими керує Spring, інформацію про середовище, в якому запущена програма (назва операційної системи, версія Java тощо).

2.1.4 База даних

База даних – організована структура, яка призначена для збереження та змін інформації. Бази даних використовуються в місцях, де інформація не є статичною, а змінюється з часом, додаються нові сутності для маніпуляцій (приклад – інтернет-магазин, який повинен зберігати користувачів, товари, ціни та багато іншого). На сьогоднішній день база даних присутня майже в кожному застосунку, а програми для автоматизації роботи з базами даних є дуже популярними серед розробників. Бази даних

Система керування базами даних (СКБД) – набір програм, що містить власне базу даних та засоби для доступу до неї. Надає можливість зберігати, оновлювати та шукати інформацію у базі та контролювати доступ до неї. Основні функції СКБД:

- підтримка цілісності бази даних;
- підтримка транзакційності та багатопоточного доступу до даних;
- встановлення певних обмежень для деяких даних (первісні ключі, зовнішні ключі, перевірка умов при додаванні нових даних).

Системи керування базами даних засновані на моделях баз даних – певних структурах для обробки даних. Кожна СКБД створюється для роботи з однією з них з урахуванням особливостей операцій над інформацією. Два найпопулярніших рішення на сьогоднішній день – це реляційна модель, в якій дані організовані у вигляді набору таблиць, що складаються з рядків та стовпців та існує можливість створювати зв’язки між ними, та безмодельний(NoSQL) підхід, який полягає у відмові від обмежень, що накладаються при використанні реляційної моделі. Даний проект буде розроблений з використанням традиційного реляційного підходу, оскільки для вирішення поставленого завдання буде необхідно створювати зв’язки між сутностями(приклад – зв’язок між користувачем та його курсами).

На сьогодні існує велика кількість СКБД, як комерційних, так і повністю безкоштовних. Серед комерційних можна відзначити Microsoft SQL Server та

Oracle, а серед безкоштовних – MySQL та PostgreSQL. Для виконання даного проекту буде використано дві СКБД – MySQL для розгортання на хостингу у мережі Інтернет та H2 для написання інтеграційних тестів та перевірки коректності роботи застосунку.

2.1.4.1 MySQL

MySQL – реляційна система керування базами даних, створена у 1995 році з використанням мов програмування C та C++. Довгий час розвивалася як проект з відкритим вихідним кодом, але після придбання компанії-розробника MySQL корпорацією Sun Microsystems у 2008 році, а потім Oracle у 2009, процес розробки стає все менш прозорим і більш відокремленим від спільноти open-source розробників. На даний момент серед користувачів даної СКБД можна відзначити Apple, Amazon та Google. Переваги MySQL:

- простота у використанні – MySQL є легким для встановлення, а широкий набір програм, що надають зручний графічний інтерфейс, ще більше спростять роботу з базою даних;
- безпека – існує велика кількість вбудованих функцій, що забезпечують безпечний доступ до даних;
- швидкість – спрощення деяких стандартів SQL дозволяє значно збільшити швидкодію;
- багатопоточність – присутня підтримка необмеженої кількості користувачів для одночасної роботи з базою даних.

2.1.4.2 H2DB

H2DB – система управління базами даних, написана на мові Java, перша версія була запущена у 2004 році. Може бути вбудована у Java-застосунок або використовуватися у режимі клієнт-сервера. Вона є де-факто стандартом серед in-memory баз даних, які використовуються передусім для виконання інтеграційних тестів та перевірки результатів роботи окремих компонентів застосунку, що взаємодіють з базою даних. Основним програмним API є SQL та

JDBC (Java Database Connectivity – стандарт для роботи з базами даних з використанням мови Java), але також присутня підтримка використання драйвера ODBC PostgreSQL для взаємодії з сервером PostgreSQL.

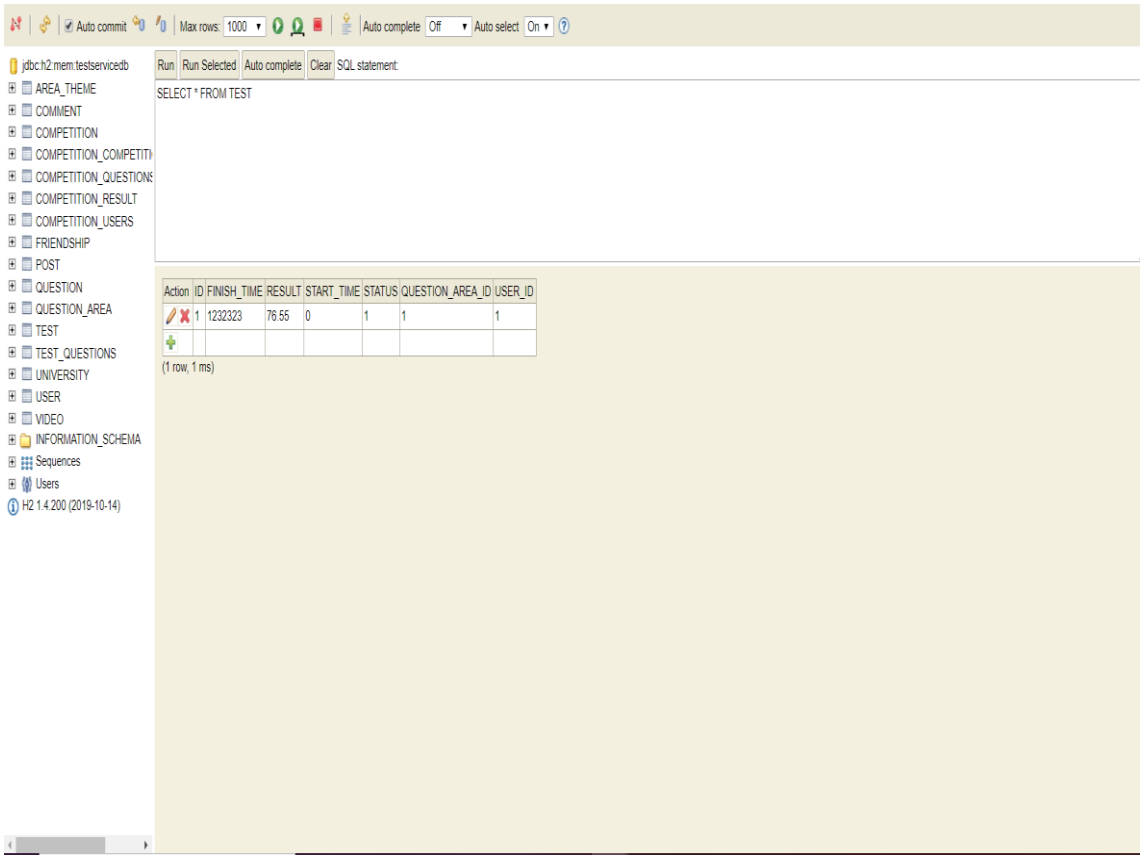


Рис. 2.1 – Зовнішній вигляд графічного інтерфейсу H2DB

Переваги H2DB:

- повна підтримка стандарту SQL;
- підтримка транзакційності;
- наявність повнотекстового пошуку;
- підтримка захисту проти SQL-ін’єкцій через використання параметризації операторів. У H2 даний функціонал називається «відключення операторів».

2.2 Клієнтська частина

Під програмуванням клієнтської частини веб-додатку зазвичай мається на увазі програмування частини застосунку, яка буде видимою для безпосереднього

користувача і до якої користувач матиме доступ. Зазвичай, ця функціональність реалізується з використанням мови програмування Javascript і результатом є HTML-сторінка у веб-браузері.

2.2.1 Загальні поняття

2.2.1.1 Браузер

Браузер – програмний застосунок для доступу до інформації з Всесвітньої павутини (World Wide Web). Перший браузер був створений у 1990 Тімом Бернерсом-Лі. Завданням браузера є отримати інформацію з мережі та відобразити її на пристрої користувача. Принцип роботи браузера полягає у під'єданні до HTTP-сервера, адреса якого вказана у адресному рядку, отриманні з нього інформації та вивід її на екран без змін(якщо було повернуто HTML-сторінку) або після виконання з нею певних дій (форматування тощо). Наразі на ринку присутні багато програмних рішень, які надають дані послуги, найпопулярнішими з них є Google Chrome, Mozilla Firefox, Safari та Internet Explorer. Ці браузери відрізняються корпораціями, які займаються їх розробкою, а також набором функцій, які ними підтримуються. Це означає, що деякі браузери можуть не підтримувати найновіші версії HTML, CSS, JavaScript та їх фреймворків аж до оновлення версії браузера або не підтримувати взагалі, тому при розробці веб-застосунку необхідно потурбуватися про те, щоб увесь необхідний функціонал коректно відображався навіть у старих браузерах.

2.2.1.2 HTML

HTML (англ. Hypertext Markup Language – мова розмітки гіпертексту) – мова, розроблена для відображення документів у веб-браузері. HTML-код найчастіше міститься у файлах з розширенням .html і складається з спеціальних семантичних елементів – дескрипторів, які також часто називають тегами. Наприкад, тег <p> відобразить абзац з текстом, що знаходиться між відкриттям дескриптора та його закриттям, тег <h1> відобразить заголовок (текст за замовчуванням буде відображений великим шрифтом та жирним). На сьогодні

основною версією HTML є HTML 5, яка була представлена у 2014 році. Використання даної версії, окрім відображення гіпертексту, надає можливість переглядати відеофайли (використання тегу <video>), прослуховувати аудіофайли (тег <audio>), зберігати дані в браузері, отримувати геолокацію і виконувати ще багато дій, для яких раніше необхідним було використання Javascript-скриптів або програвача Adobe Flash, який на сьогодні вже майже не використовується.

Приклад HTML-документу:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Приклад веб-сторінки</title>
 </head>
 <body>
  <h1>Заголовок</h1>
  <!-- Коментар -->
  <p>Перший абзац.</p>
  <p>Другий абзац.</p>
 </body>
</html>
```

Результатом, що виведеться в екран браузера, буде наступна сторінка:

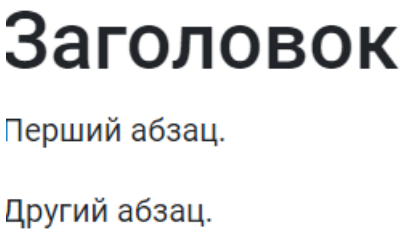


Рис. 2.2. Приклад HTML-сторінки

2.2.1.3 CSS

CSS (англ. Cascading Style Sheets – Каскадні таблиці стилів) – мова для опису зовнішнього вигляду сторінок, що написані мовою HTML. CSS було розроблено для відділення логіки, що відображається на сторінці, від опису графічних елементів, які впливають на те, як саме вона відображатиметься (кольори, шрифти, позиціонування на сторінці). CSS має досить простий синтаксис, в якому використовується лише декілька слів англійської мови, які описують імена різних властивостей стилів. Зазвичай css-код зберігається у файлах з розширенням .css, у яких міститься список правил, кожне з яких один або більше селектор, в якому описано, які саме стилі мають бути застосовані до яких елементів сторінки. Селектори можуть бути застосовані до:

- усіх елементів певного типу (наприклад, до усіх заголовків другого рівня <h2>;
- до елементів, які містять вказаний атрибут (всього існує два атрибути – id – унікальний ідентифікатор елемента на сторінці, та class – ідентифікатор, що може бути застосований до багатьох елементів на сторінці);
- до елементів, залежно від того, як вони розташовані відносно інших елементів на сторінці(наприклад, усі заголовки третього рівня <h3>, що знаходяться всередині елементу списку <li>).

2.2.1.4 AJAX

Як вже було вказано, для програмування логіки на стороні клієнта використовується мова Javascript. Але навіть так для того, щоб надіслати нові дані на сервер або отримати оновлення існуючих даних, необхідно оновлювати сторінку, що є незручним для користувача. Для вирішення цієї проблеми було розроблено підхід до побудови користувацьких інтерфейсів, за якого сторінка сама у фоновому режимі надсилає запити по HTTP до сервера і оновлює лише частину HTML-сторінки, яка повинна змінитися, базуючись на отриманих даних. Даний підхід отримав назву AJAX(Asynchronous JavaScript And XML). Важливо

ще раз зазначити, що це не окрема технологія, а саме підхід, який реалізовується з використанням цілої групи технологій, а саме:

- HTML та CSS для представлення даних на екрані користувача
- Документно-об’єктна модель(представлення HTML-документу у вигляді дерева, де кожен вузол представляє певну частина документу)
- XML (нині – майже всюди JSON) для обміну даними з сервером
- XMLHttpRequest – Javascript об’єкт, який реалізує асинхронну комунікацію з сервером
- Javascript, який об’єднує всі ці технології разом.

Узагальнимо основні плюси використання AJAX:

- можливість створення зручного для використання інтерфейсу користувача;
- легше здійснюється активна взаємодія з користувачем;
- відбувається часткове оновлення сторінки замість повного;

Незважаючи на те, що даний підхід став майже стандартом для розробки веб-застосувань, необхідно пам’ятати про певні обмеження, що накладаються його використанням:

- не працюватиме, якщо вимкнути використання Javascript у браузері;
- за ненадійних інтернет-з’єднань існує ймовірність втратити прогрес роботи зі сторінкою, оскільки може існувати часта необхідність перезавантажувати усю сторінку замість лише її частин, як того вимагає AJAX;

2.2.2 Обґрунтування обраних технологій

На сьогодні для створення клієнтської частини веб-застосунків використовують один з багаточисельних фреймворків, заснованих на мові Javascript (так званий «чистий» Javascript, тобто використання лише вбудованих можливостей мови, вже майже не використовується). Трьома найпопулярнішими з них є Angular, React та Vue.js. Кожне з цих рішень може бути застосоване для

створення проекту, який є завданням даної роботи, але мною буде використаний Angular через декілька причин:

- в основі фреймворку лежить мова Typescript – самостійна мова програмування, яка при виконанні компілюється у Javascript, але при цьому на відміну від нього підтримує повноцінне об’єктно-орієнтоване програмування, модульність та статичну типізацію;
- даний фреймворк забезпечує максимальну гнучкість при розробці та максимальну швидкість візуалізації сторінки.

2.2.3 Angular

Angular (зазвичай мається на увазі Angular версії 2 та вище) – фреймворк з відкритим кодом для створення клієнтської частини веб-застосунку. Розробляється на мові Typescript під керівництвом команди розробників з компанії Google. Є покращеною та переосмисленою версією більш старого фреймворку AngularJS, який був написаний тією ж командою розробників. Даний фреймворк з часу своєї появи у 2016 році постійно оновлюється і наразі найновішою є версія Angular 9. Незважаючи на широке використання Angular у проектах різної складності по всьому світу, основні надії на його розвиток пов’язані з його значною інтеграцією у проекти розробника фреймворка – Google, який вже представив план з довготривалої підтримки даної платформи. Angular надає розробникам та користувачам низку переваг, які вигідно виділяють його на фоні, наприклад, фреймворку React, який заснований та розвивається іншою корпорацією-гігантом Facebook, або на фоні старої версії AngularJS. Переваги Angular:

- архітектура, заснована на використанні компонентів, що підвищує якість коду та облегшує підтримку. Angular-компоненти – це маленькі компоненти користувацького інтерфейсу застосунку, кожен з яких можна розробляти окремо від інших, перевикористовувати та покривати модульними тестами

- використання Typescript, що спрощує навігацію за проектом, пошук типових помилок, які зустрічаються Javascript-розробникам, додає нові можливості з рефакторингу застосунку та додавання нових функціональних можливостей

- використання RxJS – бібліотеки для асинхронної роботи з даними, що дозволяє використовувати усі переваги асинхронної моделі програмування;

- використання спрощеної версії шаблону проектування MVC, що також позначається на якості коду та його підтримці

- використання моделі впровадження залежностей за таким самим принципом, як і фреймворк Spring, що був описаний вище, що дозволяє зберігати слабку зв'язність між різними частинами застосунку

- висока швидкість роботи застосунку, що пов'язана передусім з Ivy – розумним компілятором, який перетворює компоненти та шаблони на Javascript та HTML, а також виключає з компіляції інструкції, які не можуть бути використані

- докладна документація, викладена на офіційному сайті фреймворку разом з навчальними статтями та відео для початківців;

- велика спільнота користувачів фреймворку, які поширюють відповіді на найпоширеніші запитання з приводу роботи фреймворку.

Недоліки даної платформи не такі значні, як переваги, але також заслуговують на згадку:

- складність переходу з AngularJS на Angular, якщо виникла необхідність перейти з старої версії фреймворку на нову, це неможливо зробити без додаткових налаштувань і суттєвих змін програмного коду. Для переходу між версіями Angular є неактуальним

- складність входу у технологію для новачків – набір необхідних базових знань є більшим, ніж для React чи Vue.js, також обов'язковим є розуміння концепцій асинхронного програмування для роботи з бібліотекою RxJS, яка є потужним інструментом, але потребує певних базових знань.

Висновки до розділу 2

В даному розділі було описано основні поняття, пов’язані з розробкою серверної та клієнтської частин програмного застосунку, було обґрунтовано вибір технологій, описано їх можливості, переваги та певні обмеження.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

З точки зору архітектури продукту його можна розділити на три частини: база даних, серверна частина, яка отримує дані з бази та передає їх на клієнтську частину, завданням якої є відображення даних на екрані користувача. Розглянемо ці три компоненти по черзі.

3.1 База даних

Під час виконання даної роботи було використано реляційну СУБД MySQL. При розробці було створено 15 таблиць, перелік яких з коротким описом наведено у таблиці 3.1, детальний опис – у таблицях, починаючи з 3.2:

Таблиця 3.1

Назва таблиці	Опис
AREA	Опис навчальної дисципліни
AREA_THEME	Список тем кожної навчальної дисципліни
COMMENT	Коментарі користувачів під навчальними відео
COMPETITION	Опис змагань, в яких можуть брати участь користувачі
COMPETITION_QUESTIONS	Зв’язки між змаганнями та тестовими запитаннями, які задавались учасникам змагання
COMPETITION_RESULT	Результати проходження змагань користувачами
COMPETITION_USERS	Зв’язки між змаганнями та користувачами
FRIENDSHIP	Список друзів користувачів

Таблиця 3.1 (продовження)

Назва таблиці	Опис
POST	Список постів користувачів на власній сторінці
QUESTION	Запитання, які можуть з'являтися у тестах
TEST	Тести, які проходяться користувачами
TEST_QUESTIONS	Зв'язок між запитаннями та тестами, у яких вони з'являються
UNIVERSITY	Університети, посилання на які є в системі
USER	Дані про користувачів системи
VIDEO	Відео, які відображаються у процесі проходження курсів

Опис таблиці AREA

Таблиця 3.2

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ – унікальний ідентифіктор запису в таблиці
IMAGE_URL	VARCHAR (255)	Посиланяя на картинку, що пов'язана з даною навчальною дисципліною
TITLE	VARCHAR (255)	Текстовий опис навчальної дисципліни

Опис таблиці AREA_THEME

Таблиця 3.3

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
THEME	VARCHAR (255)	Текстовий опис теми
AREA_ID	BIGINT	Посилання на навчальну дисципліну(таблиця QUESTION_AREA)

Опис таблиці COMMENT

Таблиця 3.4

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
TEXT	VARCHAR (1023)	Зміст коментаря
TIME	BIGINT	Час написання коментаря, зберігається у мілісекундах, починаючи з 1 січня 1970 року
USER_ID	BIGINT	Посилання на користувача, що залишив коментар (таблиця USER)
VIDEO_ID	BIGINT	Посилання на відео, під яким залишено коментар (таблиця VIDEO)

Опис таблиці COMPETITION

Таблиця 3.5

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
START_TIME	BIGINT	Час початку змагання у мілісекундах
STATUS	TINYINT	Статус змагання (ще продовжується або вже закінчене)
AREA_ID	BIGINT	Посилання на навчальну дисципліну, з якої проводиться змагання (таблиця AREA)

Опис таблиці COMPETITION_RESULT

Таблиця 3.6

Назва поля	Тип у базі даних	Опис
COMPETITION_ID	BIGINT	Частина первісного ключа, посилання на змагання (таблиця COMPETITION)
USER_ID	BIGINT	Частина первісного ключа, посилання на користувача (таблиця USER)
RESULT	DECIMAL (5, 2)	Результат змагання
TIME_SPENT	BIGINT	Витрачений час у мілісекундах

Опис таблиці FRIENDSHIP

Таблиця 3.7

Назва поля	Тип у базі даних	Опис
ACCEPTED_USER_ID	BIGINT	Частина первісного ключа, посилання на користувача, що відповідає на запит про дружбу (таблиця USER)
INVITED_USER_ID	BIGINT	Частина первісного ключа, посилання на користувача, що надіслав запит про дружбу (таблиця USER)
INVITE_TIME	BIGINT	Час надсилання запиту у мілісекундах
RESPONSE_TIME	BIGINT	Час відповіді на запит у мілісекундах
STATUS	TINYINT	Статус (очікує на відповідь, прийняти запит або відмовлено)

Опис таблиці POST

Таблиця 3.8

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
TEXT	VARCHAR (1023)	Текст посту користувача
TIME	BIGINT	Час викладення у мілісекундах

Таблиця 3.8 (продовження)

Назва поля	Тип у базі даних	Опис
USER_ID	BIGINT	Посилання на користувача, що публікує пост (таблиця USER)

Опис таблиці QUESTION

Таблиця 3.9

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
ANSWER_OPTIONS	TEXT	Текстовий рядок у форматі JSON, що містить варіанти відповіді на запитання і вказання правильної відповіді
TEXT	VARCHAR (1023)	Текст запитання
TYPE_ID	TINYINT	Вказує на тип запитання (з однією відповіддю, з багатьма відповідями або відкрите)
AREA_THEME_ID	BIGINT	Посилання на тему, до якої належить запитання (таблиця AREA_THEME)

Опис таблиці TEST

Таблиця 3.10

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
FINISH_TIME	BIGINT	Час закінчення тесту у мілісекундах
RESULT	DECIMAL (5, 2)	Результат проходження тесту
START_TIME	BIGINT	Час початку тесту
STATUS	TINYINT	Статус (триває або закінчився)
AREA_ID	BIGINT	Посилання на навчальну дисципліну, з якої проводився тест (таблиця AREA)
USER_ID	BIGINT	Посилання на користувача, який проходив тест (таблиця USER)

Опис таблиці UNIVERSITY

Таблиця 3.11

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
NAME	VARCHAR (255)	Назва
URL	VARCHAR (255)	Посилання на офіційний сайт університету

Опис таблиці USER

Таблиця 3.12

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
EMAIL	VARCHAR (255)	Електронна пошта користувача
ENABLED	BOOLEAN	Прапор, який вказує, чи активований акаунт користувача
GRADUATION_YEAR	SMALLINT	Рік випуску користувача зі школи, для легшого пошуку знайомих
LOGIN	VARCHAR (255)	Унікальне ім’я користувача на ресурсі
PASSWORD	VARCHAR (255)	Зашифрований пароль, за яким здійснюється авторизація
REAL_NAME	VARCHAR (255)	Справжнє ім’я користувача
ROLE	TINYINT	Роль користувача (студент, викладач, адмін)
DESIRED_UNIVERSITY_ID	BIGINT	Бажаний університет для вступу, для легшого пошуку знайомих

Опис таблиці VIDEO

Таблиця 3.13

Назва поля	Тип у базі даних	Опис
ID	BIGINT	Первісний ключ
URL	VARCHAR (255)	Посилання на зовнішній ресурс, з якого необхідно завантажити відео
AREA_THEME_ID	BIGINT	Посилання на навчальну тему, до якої належить відео (таблиця AREA_THEME)

3.2 Серверна частина

При розробці серверної частини продукту було використано такі технології:

- Spring Boot
- Spring Data JPA
- Spring MVC
- Spring Security
- Maven – засіб для автоматизації будування проекту
- JUnit – бібліотека для написання модульних тестів
- Mockito – бібліотека для створення заглушок реальних об’єктів для тестування бібліотекою JUnit
- в якості середовища розробки - Intellij Idea.

Розглянемо структуру розробленого проекту:

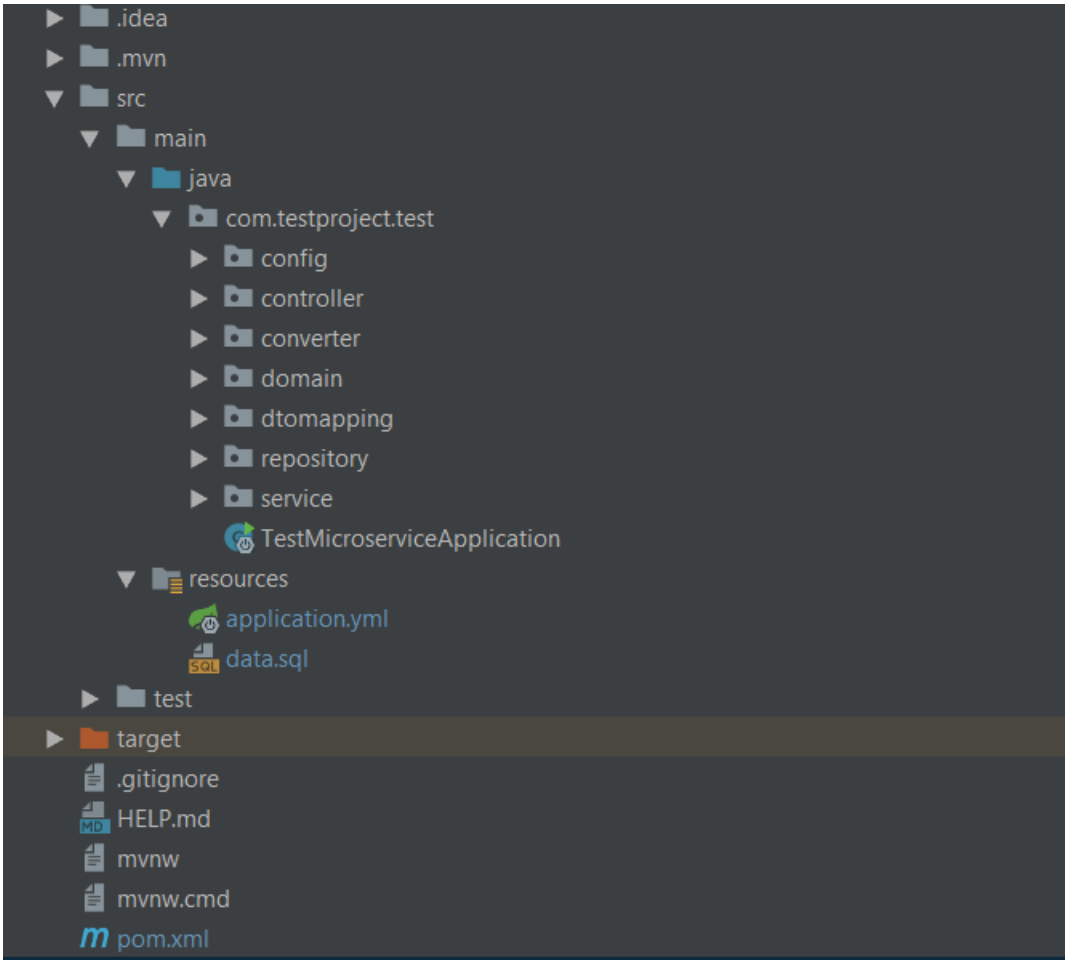


Рис. 3.1 – Структура серверної частини проекту

Кожна з папок містить набір класів, що відповідають за реалізацію певного функціоналу, за яким вони і згруповані. Розглянемо їх детальніше.

Папка /config містить класи, в яких перевизначаються стандартні конфігурації, створені Spring Boot. Тут визначаються параметри безпеки для застосунку, наприклад, до яких URL можна отримати доступ без попередньої авторизації, а також конфігуруються параметри CORS, за допомогою яких клієнтська частина застосунку отримає можливість комунікувати з серверною (у стандартній конфігурації дана можливість вимкнена).

У папці /converter знаходяться конвертери, які перевизначають поведінку Spring Data фреймворку з перетворення деяких полів таблиць бази даних на поля Java-об’єктів.

У папці /domain знаходяться усі класи, які представляють таблиці бази

даних (наприклад, клас User, який є відображенням таблиці бази USER) та допоміжні сутності (наприклад, enum FriendshipStatus, який відображає поле STATUS таблиці FRIENDSHIP). Для того, щоб Java-клас був розпізнаний як представлення таблиці бази даних, над назвою класу ставиться анотація @Entity. Також за допомогою анотацій вказуються зв'язки з іншими таблицями та обмеження, що накладаються на поле таблиці у базі даних.

```java
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Size(min = 10, max = 255)
    @Column(unique = true, nullable = false)
    private String login;

    @Size(min = 10)
    @Column(nullable = false)
    private String password;

    @Email
    @Column(unique = true)
    private String email;

    private Role role;

    private String realName;

    private Integer graduationYear;

    @ManyToOne
    private University desiredUniversity;

    private boolean enabled;

    @OneToMany(mappedBy = "user")
    private List<Test> tests;

    @OneToMany(mappedBy = "user")
    private List<Post> posts;

    @OneToMany(mappedBy = "invitedUser", cascade = CascadeType.ALL)
    private List<Friendship> invitedFriendships;

    @OneToMany(mappedBy = "acceptedUser", cascade = CascadeType.ALL)
    private List<Friendship> acceptedFriendships;

    public User(Long id) { this.id = id; }
}
```

Рис. 3.2 – Приклад класу, що представляє сутність БД

Папка /dtomapping містить так звані DTO-класи, які не зберігаються у базу даних, але представляють інформацію, яку необхідно видати користувачу як відповідь на запит або прийняти від користувача у тілі запиту.

Папка /controller містить набір контролерів, які приймають HTTP-запити користувачів, передають їх на обробку, а потім повертають користувачу результат виконання запиту. Кожен контролер помічений анотаціями

@RestController та @RequestMapping. Анотація @RestController вказує на те, що результатом виконання даного запиту буде не HTML-сторінка, а текст. Анотація @RequestMapping містить параметр, який вказує URL, який має оброблятися даним контролером.

Папка /repository містить клас-репозиторії, які відповідають за роботу з базою даних. Для кожної сутності, яка знаходиться у папці /domain створюється клас НазваСутностіRepository (наприклад, UserRepository), через який виконується робота з цією сутністю у базі даних. Цей клас вже містить набір базових методів, наприклад, save для зберігання сутності у базі, або findAll для отримання списку усіх записів, які присутні у базі. Також можливо додавати власні запити, написані мовою SQL, JPQL або взагалі просто вказавши назву методу відповідно до специфікації фреймворку Spring Data. Наприклад, метод findByLogin при виклику поверне результат запиту select u from user u where u.login = параметр.

```
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByLoginOrEmailAndEnabledTrue(String login, String email);

    @Query("select case when count(u)> 0 then true else false end from User u " +
            "where u.login = :login or u.email = :email")
    boolean existsByLoginOrEmail(@Param("login") String login, @Param("email") String email);

    Optional<User> findByLogin(String login);

    @Query("select u from User u left join fetch u.tests where u.id = :id")
    Optional<User> findByIdWithAllInfo(@Param("id") Long id);

    @Query("select u from User u where " +
            "lower(u.login) like lower(concat('%', concat(:pat, '%'))) or " +
            "lower(u.realName) like lower(concat('%', concat(:pat, '%')))")
    List<User> findAllByNameOrLoginLike(@Param("pat") String pattern);
}
```

Рис. 3.3 – Приклад класу-репозиторію

Папка /service містить основну логіку застосунку, яка найчастіше полягає у отриманні певних даних з бази, проведення деяких розрахунків та їх перетворення у вигляд, в якому їх необхідно передати користувачу.

Папка /resources містить набір ресурсних файлів, наприклад, файл з властивостями проекту (там повинні знаходитися усі константи, які використовуються при роботі всього проекту, параметри налаштування

інтеграції з зовнішніми системами, наприклад, з базою даних, параметри логування).

Рис. 3.4 – Файл з усіма властивостями проекту

Папка /test містить набір класів, які тестують основну логіку застосунку для легшого внесення змін та контролю за якістю роботи програми.

Папка /target містить зкомпільовані java-класи, які і виконуються при роботі застосунку.

3.2.1 Безпека застосунку

Для налаштування безпеки системи і збереження комунікації між серверною та клієнтською частинами було використано модель, засновану на JWT-токенах. Вона полягає у передачі разом з HTTP-запитом спеціального заголовку, який містить рядок з зашифрованими іменем користувача, його правами та датою закінчення дії цього токену. Цей рядок розшифровується при надходженні на сервер, внаслідок чого користувачу надається або забороняється доступ до певної інформації.

Приклад роботи в розробленому застосунку: при спробі отримати доступ до сторінки /friends неавторизований користувач надішле на сервер запит без

авторизаційного заголовка і буде перекинутий на сторінку, на якій здійснюється ввід логіну та паролю для входу в систему.

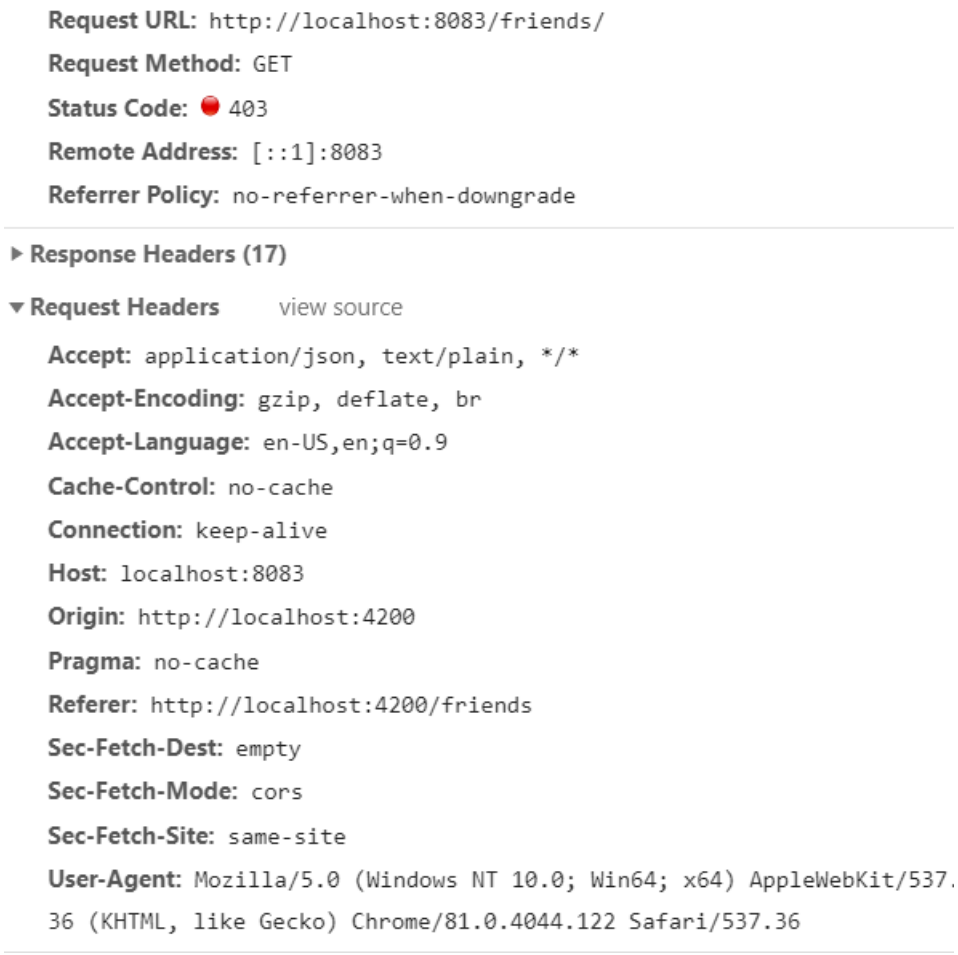


Рис. 3.5 – Відповідь сервера для неавторизованого користувача

Після проведеної процедури авторизації в локальному сховищі браузера зберігається авторизаційний токен, який передається при кожному запиті у заголовку Authorization: Bearer ТОКЕН. У консолі розробника в браузері той самий запит виглядатиме вже так:

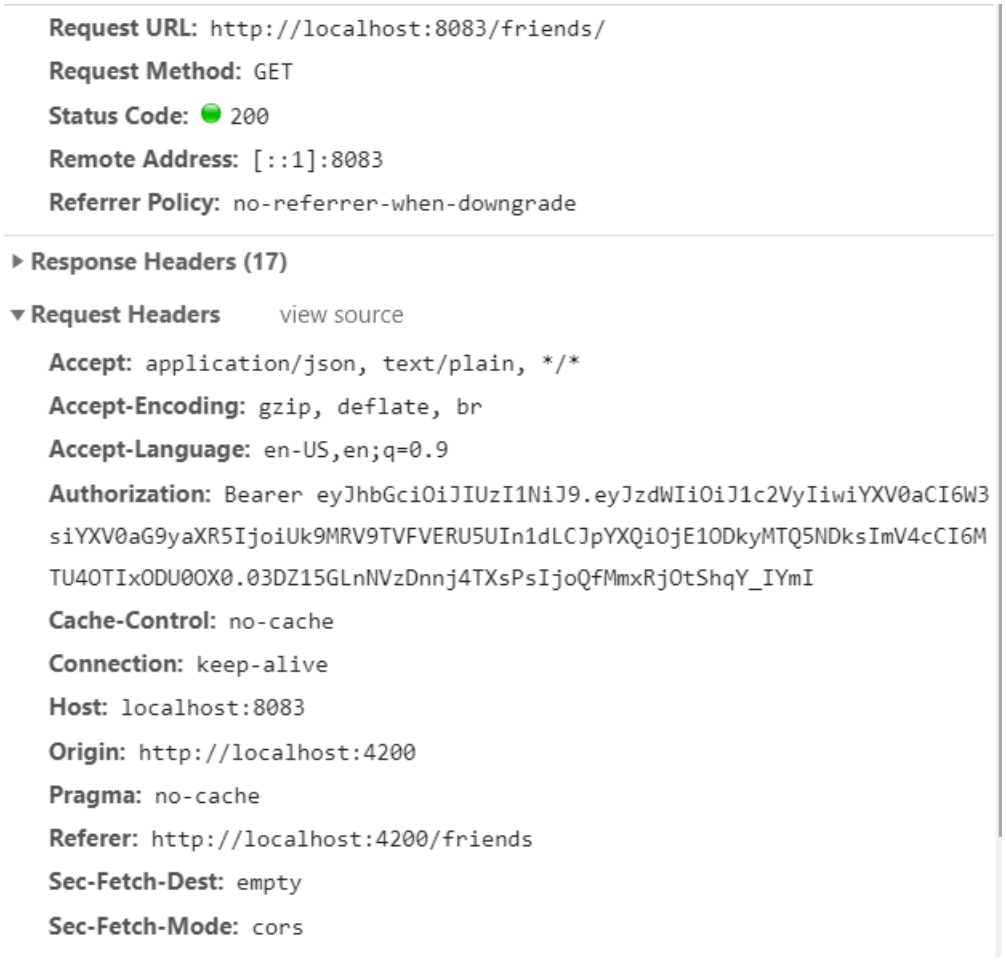


Рис. 3.6 – Відповідь сервера для авторизованого користувача

3.3 Клієнтська частина

Для створення клієнтської частини програмного застосунку було використано такі технології:

- HTML
- CSS
- Bootstrap (набір стилів для HTML)
- Angular

Розглянемо структуру проекту:

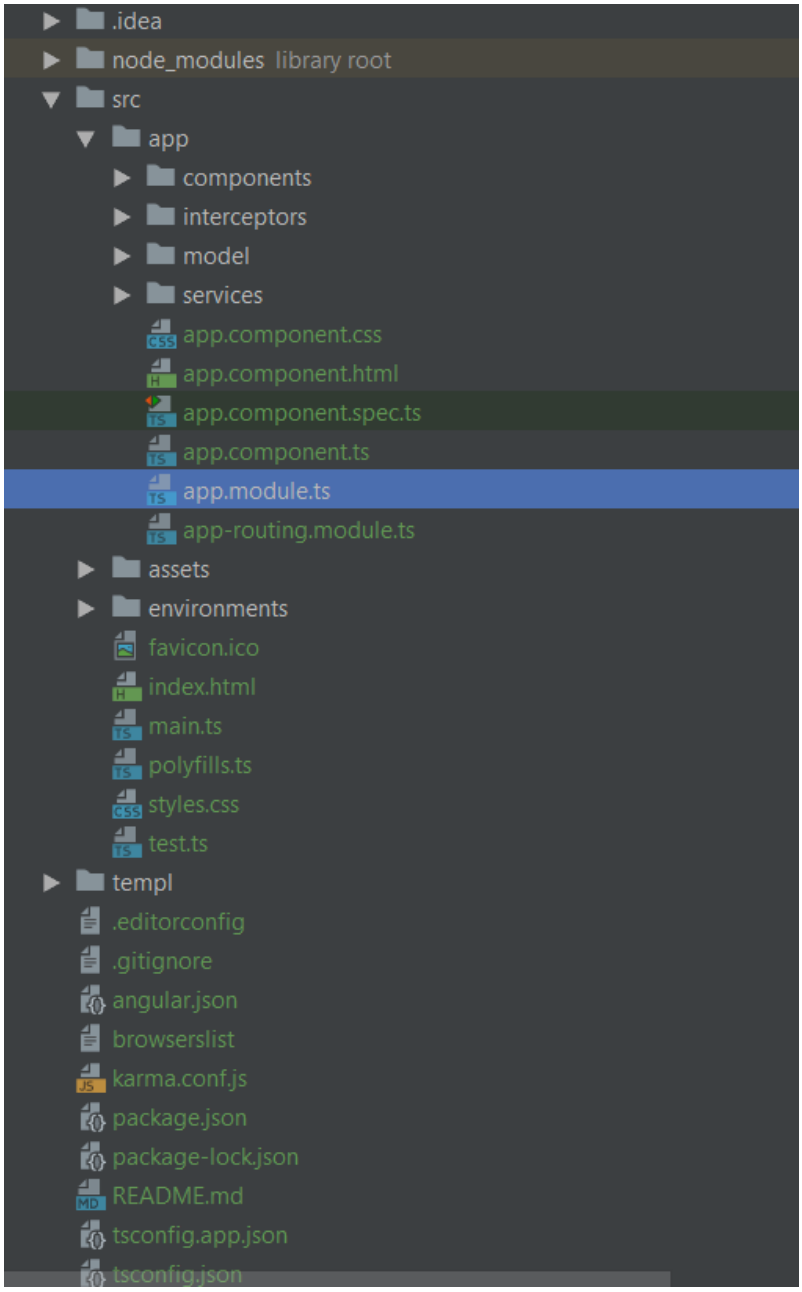


Рис. 3.7 – Структура клієнтської частини проекту

Усі компоненти проекту, крім вмісту папки /app є згенерованими автоматично інтегрованим середовищем розробки при створенні проекту, там знаходяться базові файли для роботи проекту, наприклад, список залежностей (файл package.json) та аналізатор коду, який перевіряє його на відповідність загальновживаним стандартам (файл tslint.json).

Розглянемо детальніше вміст папки /app. У папці /components знаходяться компоненти, які керують відображенням інформації на екрані, всередині кожного компонента присутні файл з кодом на Typescript, в якому описана логіка

компоненту, HTML-файл, який виводиться на пристрій користувача залежно від логіки, та CSS-файл, який накладає стилі на HTML-розмітку.

У папці /interceptors знаходяться елементи, які повинні перехоплювати певні дії в роботі програми та певним чином на них реагувати. Вміст папки:

- клас JwtInterceptor, який перехоплює HTTP-запити до віддаленого сервера та додає до них Authorization заголовок з JWT-токеном (був розглянутий у пункті 3.2.1)
- клас GlobalErrorHandler, який перехоплює помилки, які виникають у системі та коректно їх обробляє (наприклад, якщо будь-який HTTP-запит повернув результат з кодом 403 Forbidden, то користувач буде перенаправлений на сторінку авторизації, де йому необхідно буде ввести логін та пароль).

Папка /model містить усі класи, які необхідні для обміну інформацією з веб-сервером, тобто ті, в яких містяться сутності, що передаються на сервер або отримуються з нього.

Папка /services містить так звані класи-сервіси, основним завданням яких є надсилання HTTP-запитів за різними адресами.

Файли app.component.* характеризують базовий компонент застосунку, в який вбудовуються всі інші. В даному проекті даний компонент допомагає групувати на одній сторінці заголовок сторінки (верхню панель), яка повинна бути присутньою на будь-якій сторінці застосунку, та тіло сторінки, яке змінюється при навігації за сайтом.

3.4 Розгортання застосунку на зовнішньому сервері

Оскільки розробка клієнтської та серверної частин була розділеною та було отримано два незалежних застосунки, які комунікують одне з одним лише через використання HTTP-протоколу, розгортати необхідно також два окремих проекти. Розгортання серверної частини можна виконувати на хмарних сервісах як-то Amazon AWS або Google Cloud, але це вимагає також базових знань роботи з хмарними технологіями, а також є небезкоштовним, тому для пробних запусків

без участі реальних користувачів та великого навантаження на систему краще використовувати безкоштовних хостинг, який надає зрозумілі інструкції з завантаження програмних застосувань – Heroku.

Для розгортання клієнтської частини розробники Angular надають докладні пояснення усього процесу на офіційному сайті платформи, там є приклади завантаження застосунку на віддалений сервер, а також на GitHub pages – веб-хостинг, який пропонується розробниками системи контролю версій GitHub для розміщення веб-додатків і є достатньо зручним у застосуванні, тому для розгортання буде використаний саме GitHub pages.

Висновки до розділу 3

В даному розділі було детально описано основні компоненти розробленого програмного рішення, описано структуру бази даних, архітектуру серверної та клієнтської частин, та досліджено механізм захисту доступу до серверної частини застосунку з використанням токенів.

РОЗДІЛ 4
ОГЛЯД РОЗРОБЛЕНОГО ПРОДУКТУ

В даному розділі буде оглянуто основний функціонал розробленого застосунку, описано навігацію між різними його компонентами та продемонстровано деякі приклади.

4.1 Головна сторінка

Точкою входу в систему є головна сторінка (рис. 4.1), яку можуть переглядати усі користувачі, незалежно від того, чи є вони авторизованими. На ній можна побачити список навчальних дисциплін, матеріали з яких вже присутні в базі даних системи.

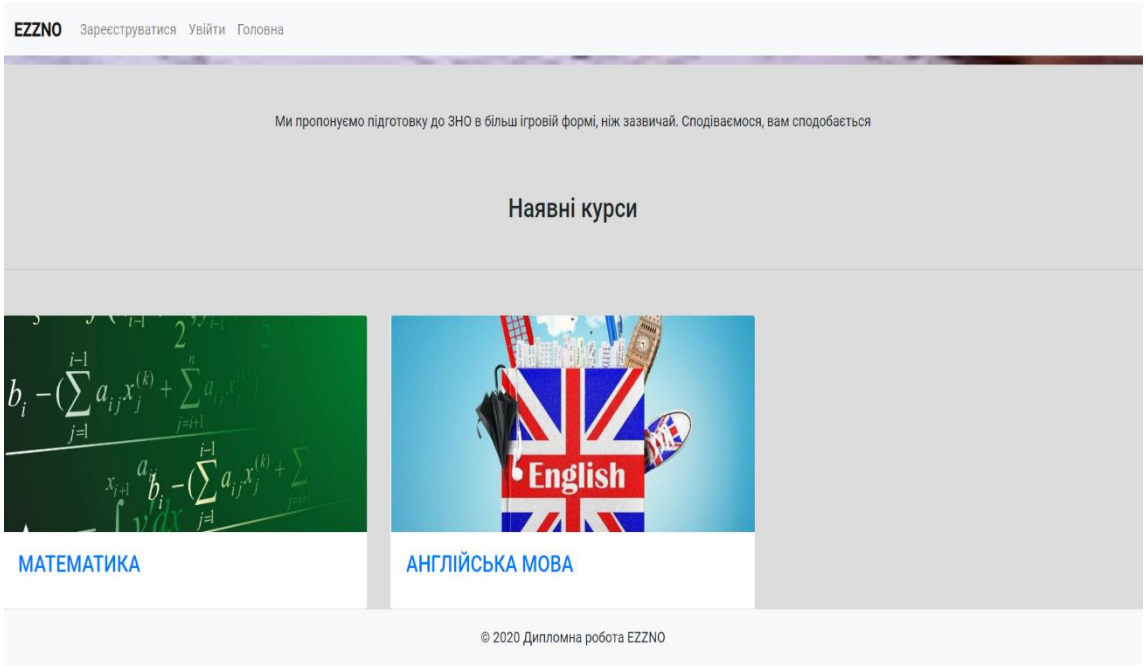


Рис. 4.1 – Головна сторінка застосунку

4.2 Реєстрація

При натисканні на кнопку «Зареєструватися» здійснюється перехід на сторінку реєстрації, де розміщена форма для вводу персональних даних (рис. 4.2). Користувачу необхідно вказати бажаний логін для авторизації, пароль, електронну пошту, роль та справжнє ім’я. При вводі неприпустимих символів, невиконанні певних вимог до даних або інших помилках, поля, які необхідно

виправити, стануть підсвіченими червоним кольором та з’явиться опис помилки. За успішної реєстрації введені логін, пароль та електронну пошту можна буде використовувати для авторизації.

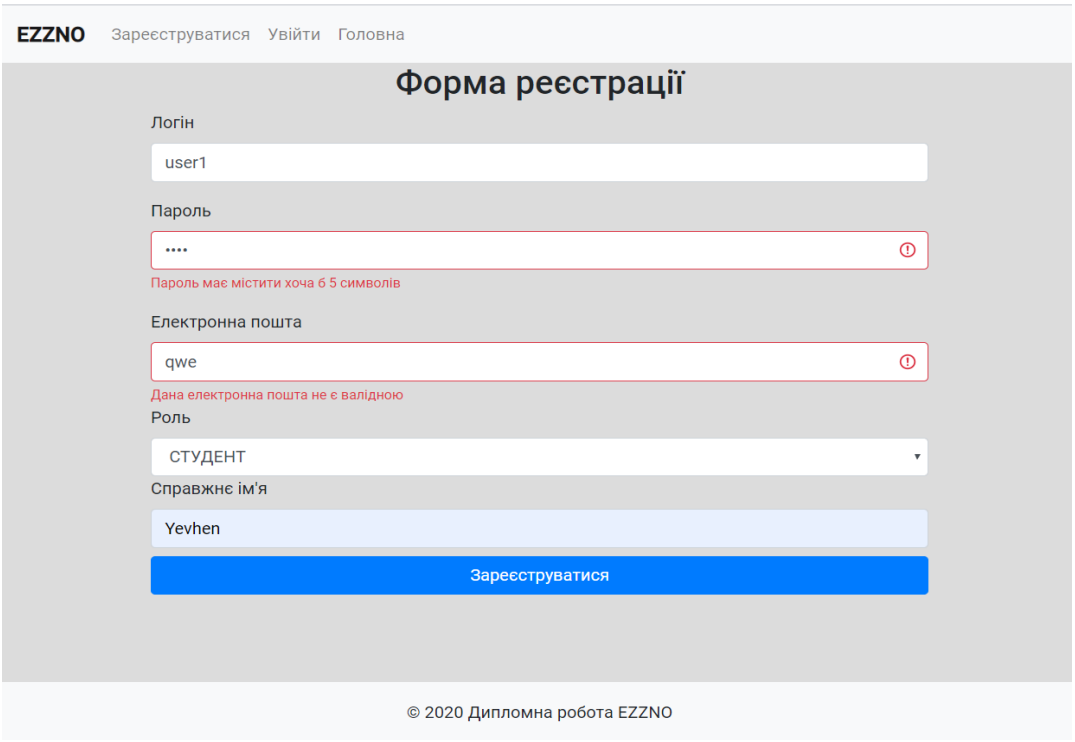


Рис. 4.2 – Сторінка реєстрації

4.3 Авторизація

При натисканні на кнопку «Увійти» здійснюється перехід на сторінку авторизації (рис. 4.3), на якій необхідно ввести логін та пароль, які використовувалися при реєстрації. У випадку вводу даних, які відсутні в базі, користувач побачить повідомлення про помилку з проханням спробувати ще раз. Якщо авторизація пройшла успішно, користувач буде переправлений на головну сторінку, але у верхній частині сторінки з’явиться ім’я користувача та список сторінок, які він може відкрити.

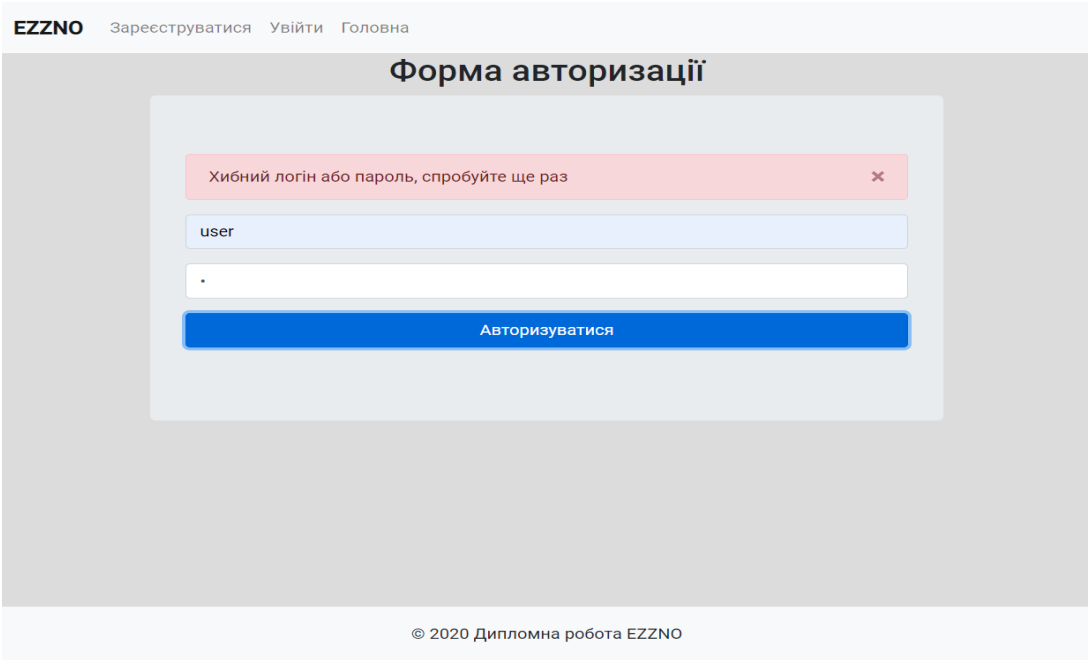


Рис. 4.3 – Сторінка авторизації

4.4 Перегляд навчальних матеріалів

Після успішної авторизації користувач отримує можливість переглядати контент застосунку, передусім, навчальні матеріали з різних дисциплін. Для цього необхідно натиснути перейти на сторінку з дисципліною, яка цікавить користувача, після чого зі списку обрати тему, матеріали з якої потрібно відкрити. В результаті отримаємо відеоряд з матеріалів, які необхідно переглянути для повного засвоєння теми. Під кожним матеріалом користувач може залишити власний відгук або поділитися чимось, що буде корисним іншим користувачам в контексті даної підтеми.

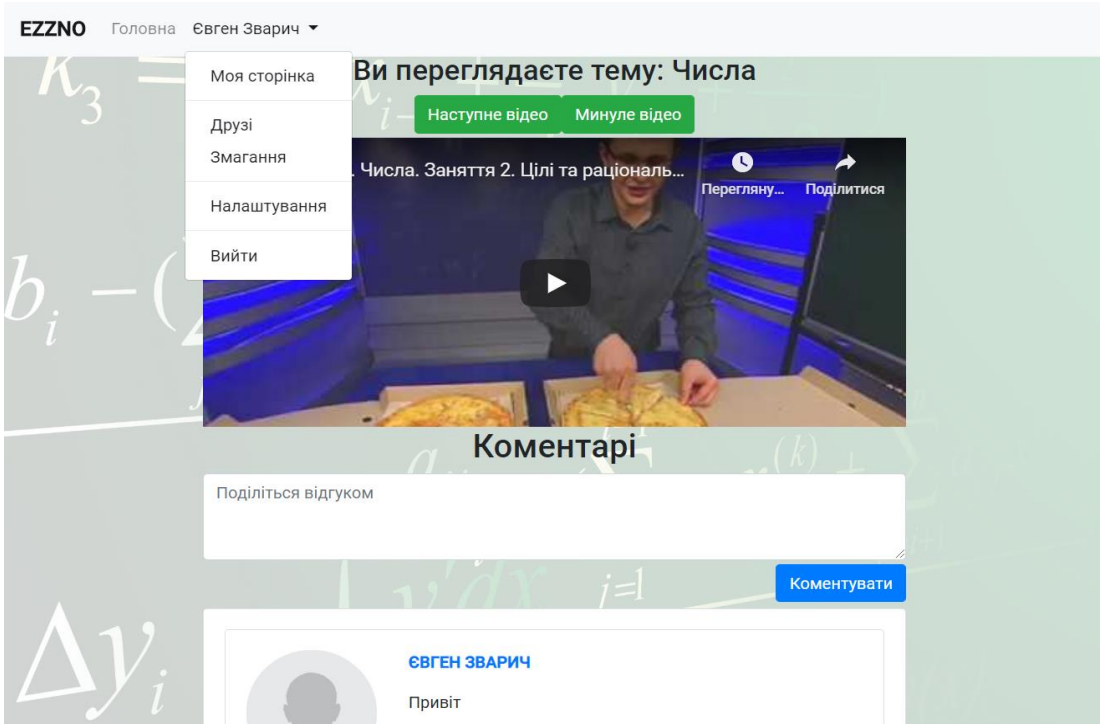


Рис. 4.4 – Сторінка з навчальними матеріалами

4.5 Проходження навчальних тестів

Для перевірки власних знань з навчальної дисципліни авторизованому користувачу необхідно перейти на сторінку дисципліни, що його цікавить, а потім натиснути кнопку «Пройти тест». Після цього користувач побачить список запитань різної форми (з однією правильною відповіддю, з багатьма, відкриті). Після закінчення тесту користувач побачить свій бал та інформацію про те, на які запитання він відповів правильно, а на які ні.

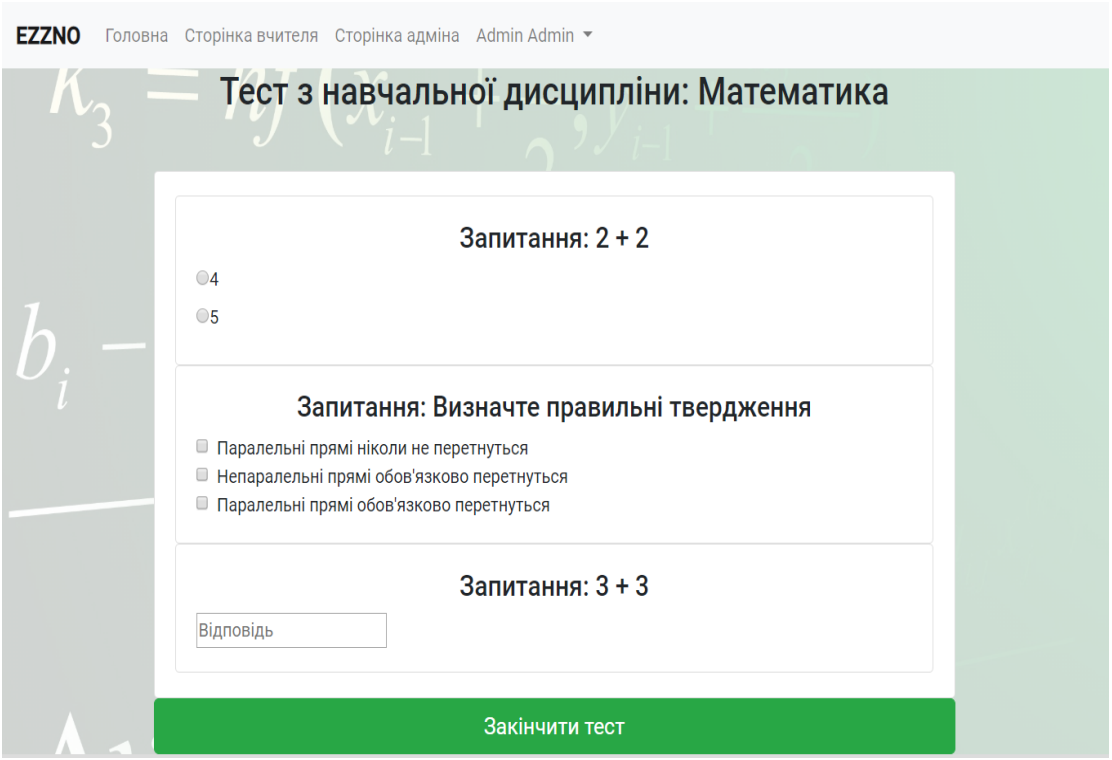


Рис. 4.5 – Сторінка проходження тесту

4.6 Перегляд інформації про користувачів

Як було згадано у розділах вище, зручна взаємодія між користувачами має бути суттєвою перевагою розробленого застосунку. Тому у системі доданий пошук користувачів за ім’ям, можливість додавати інших користувачів у друзі та переглядати їх останні оновлення. Для доступу до цього функціоналу авторизований користувач повинен натиснути кнопку «Друзі» у випадаючому меню, що відкривається при натисканні на власний логін у верхній панелі сторінки. На сторінці, що буде відкрита, користувач зможе переглянути список своїх друзів, список запрошень у друзі, на які він ще не відповів та форму вводу для пошуку інших користувачів. Сторінка для перегляду інформації про користувачів зображена на рисунку:

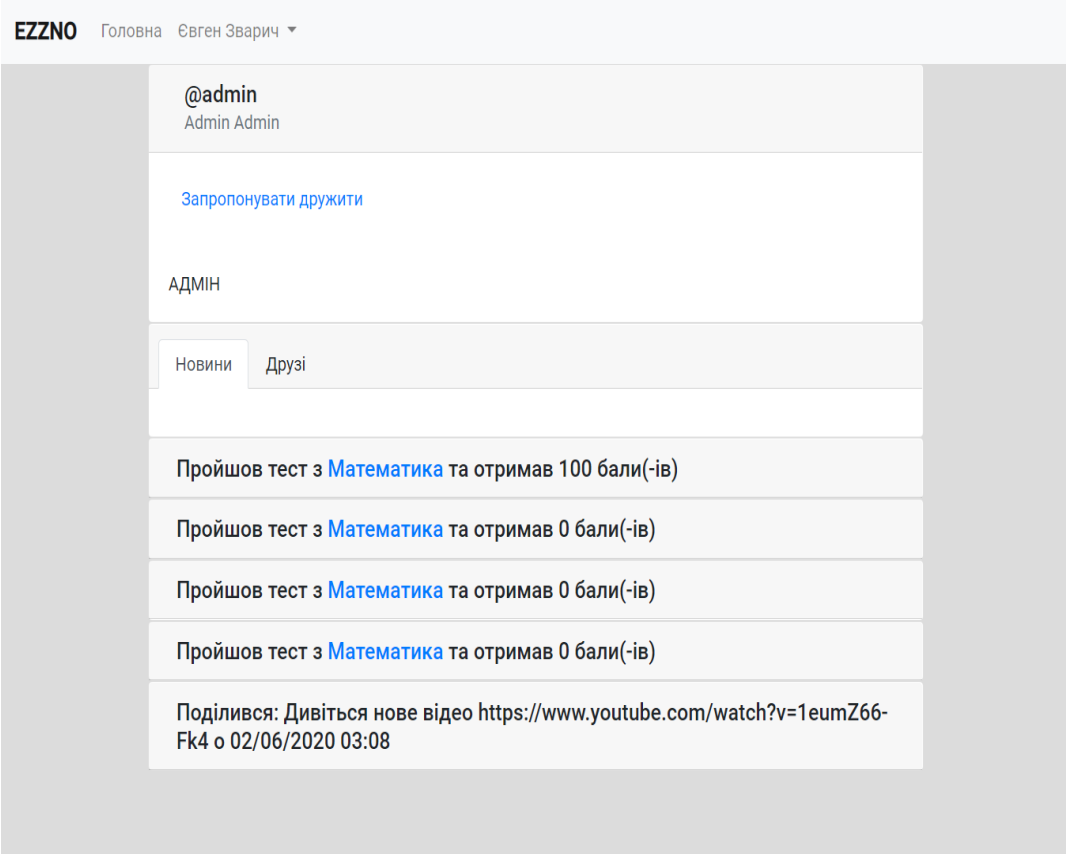


Рис. 4.6 – Сторінка користувача

4.7 Змагання між користувачами

Ще однією складовою інтерактивної взаємодії між користувачами є змагання між ними у різних навчальних дисциплінах. Для перегляду цього функціоналу авторизований користувач повинен натиснути кнопку «Змагання» у випадаючому меню, що відкривається при натисканні на власний логін у верхній панелі сторінки. Після цього він побачить меню, в якому можна переглянути майбутні змагання, змагання, в яких братиме участь користувач та створити нове змагання. Дана сторінка з відкритою формою для створення нового змагання зображена на рисунку 4.7.

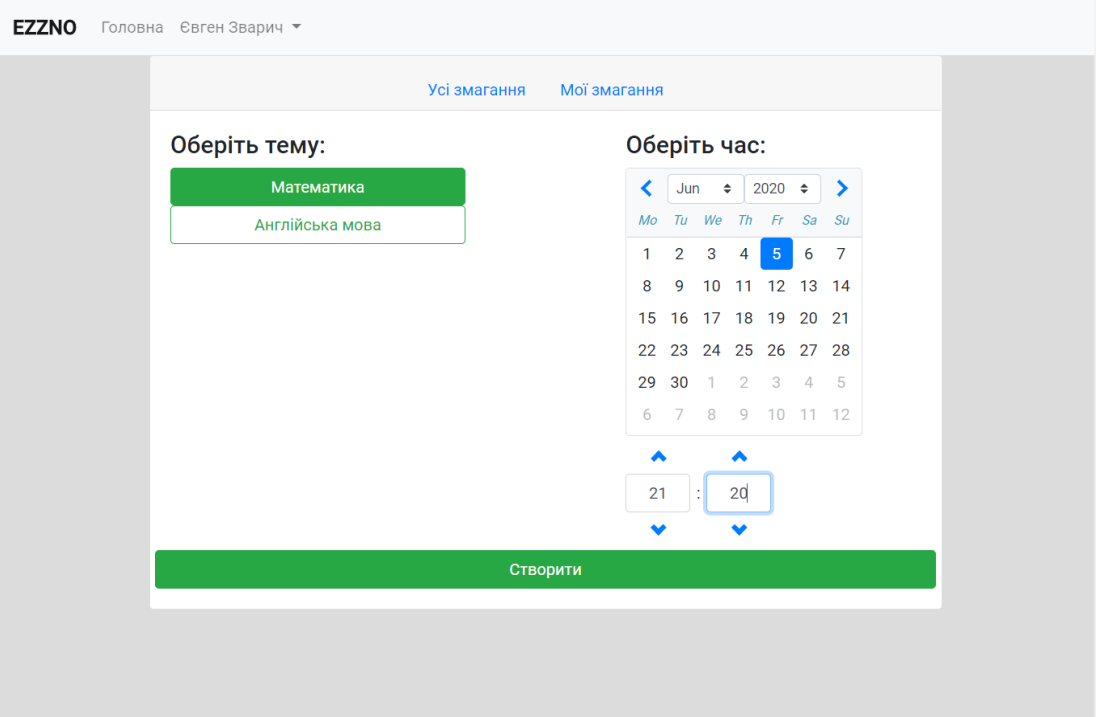


Рис. 4.7 – Сторінка створення нового змагання

Висновки до розділу 4

В даному розділі було описано основні можливості для користувача застосунку, створено мінімальну інструкцію для користувача з навігацією клієнтською частиною системи та наведено приклади графічного інтерфейсу деяких компонентів.

ВИСНОВОК

Під час виконання даної дипломної роботи було детально розглянуто проблему підготовки абітурієнтів до вступних іспитів до вищих навчальних закладів, досліджено використання онлайн-навчання як одного із засобів вирішення цієї проблеми. Також було показано декілька рішень, які так чи інакше вже вирішують поставлену проблему, розглянуто їх переваги та недоліки.

Результатом даного дипломного проекту є система для підготовки абітурієнтів до вступу, яку можна використовувати як альтернативу вже існуючим. Вона об’єднує в собі функціональні можливості наявних рішень, такі як перегляд навчальних матеріалів та проходження тестових завдань, з інтерактивним підходом до навчання, завдяки якому студенти можуть комунікувати одне з одним, брати участь у змаганнях, слідкувати за найкращими результатами, ділитися інформацією. Дана система розрахована на користувачів, які планують складати вступні іспити та викладачів, які планують допомагати їм у цьому шляхом поширення навчальних матеріалів. Незважаючи на те, що система є вже готовою до застосування, її можна доповнювати новими функціями та розширювати, щоб зробити ще більш зручною для користування.

Для створення платформи було використано сучасні технології, які є широко вживаними у галузі розробки програмного забезпечення та мають велику кількість функціональних можливостей, їх використання у даному проекті було обґрунтовано.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. ЗНО [Електронний ресурс] - Режим доступу: https://www.planetaschastya.com/sho-take-zno
2. MVC [Електронний ресурс] - Режим доступу: https://web-creator.ru/articles/mvc
3. Вища освіта в Україні [Електронний ресурс] - Режим доступу: https://osvita.ua/vnz/56653/
4. Зовнішнє незалежне оцінювання [Електронний ресурс] - Режим доступу: https://osvita.ua/test/51694/
5. Масові відкриті онлайн-курси [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/ Масові_відкриті_онлайн-курси
6. Moodle [Електронний ресурс] - Режим доступу: https://lmslist.ru/free-sdo/obzor-moodle/
7. ILearn [Електронний ресурс] - Режим доступу: https://kfund.ua/uk/projects/vseukrayinska-onlajn-platforma-ilearn/
8. Онлайн-курси [Електронний ресурс] - Режим доступу: https://studway.com.ua/5-platform-onlayn-kursiv/
9. HTTP [Електронний ресурс] - Режим доступу: https://znaimo.com.ua/HTTP
10. Веб-сервер [Електронний ресурс] - Режим доступу: http://5fan.ru/wievjob.php?id=69070
11. Spring Framework Documentation [Електронний ресурс] - Режим доступу: https://docs.spring.io/spring/docs/current/spring-framework-reference/
12. Spring Security: Authentication and Authorization In-Depth [Електронний ресурс] - Режим доступу: https://www.marcobehler.com/guides/spring-security

13. What is Spring Data JPA? And why should you use it? [Електронний ресурс] - Режим доступу: https://thorben-janssen.com/what-is-spring-data-jpa-and-why-should-you-use-it/

14. Сравнение современных СУБД [Електронний ресурс] - Режим доступу: http://drach.pro/blog/hi-tech/item/145-db-comparison

15. H2 Database Engine [Електронний ресурс] - Режим доступу: https://ru.bmstu.wiki/H2_Database_Engine

16. The Good and the Bad of Angular Development [Електронний ресурс] - Режим доступу: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/

17. Deploying Spring Boot Applications to Heroku [Електронний ресурс] - Режим доступу: https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku

18. Overview of GitHub Pages [Електронний ресурс] - Режим доступу: https://kbroman.org/simple_site/pages/overview.html

19. Angular Deployment [Електронний ресурс] - Режим доступу: https://angular.io/guide/deployment

Додатки

ДОДАТОК 1

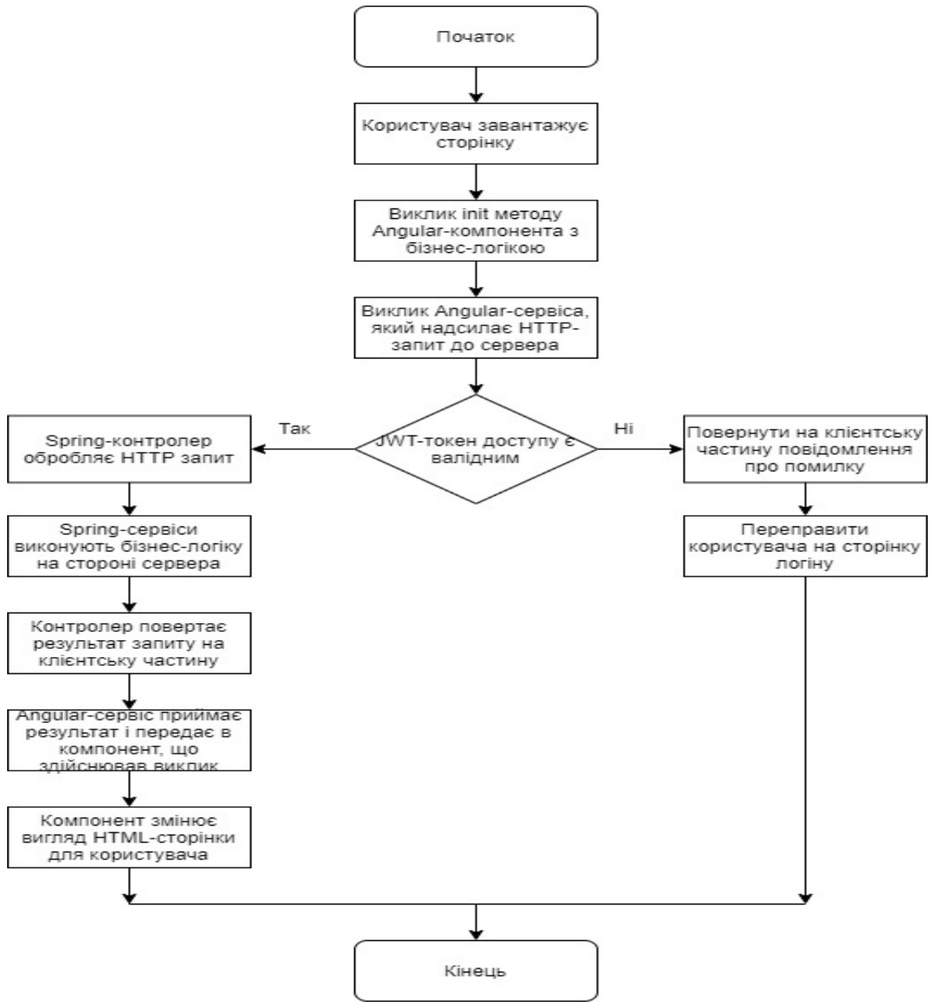
Система підготовки абітурієнтів до вступу

Схема алгоритму роботи програми

ІАЛЦ.467100.004 Д1

Аркушів 1

2020



					ІАЛЦ.467100.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата	Схема алгоритму роботи програми	Літ. / Маса / Масштаб
Розроб.		Зварич Є.О.				
Перевір.		Габінет А.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П			Дипломна робота	КПІ ім.Ігоря Сікорського, ФІОТ кафедра ОТ, гр. ІП-62
Затверд.						

ДОДАТОК 2

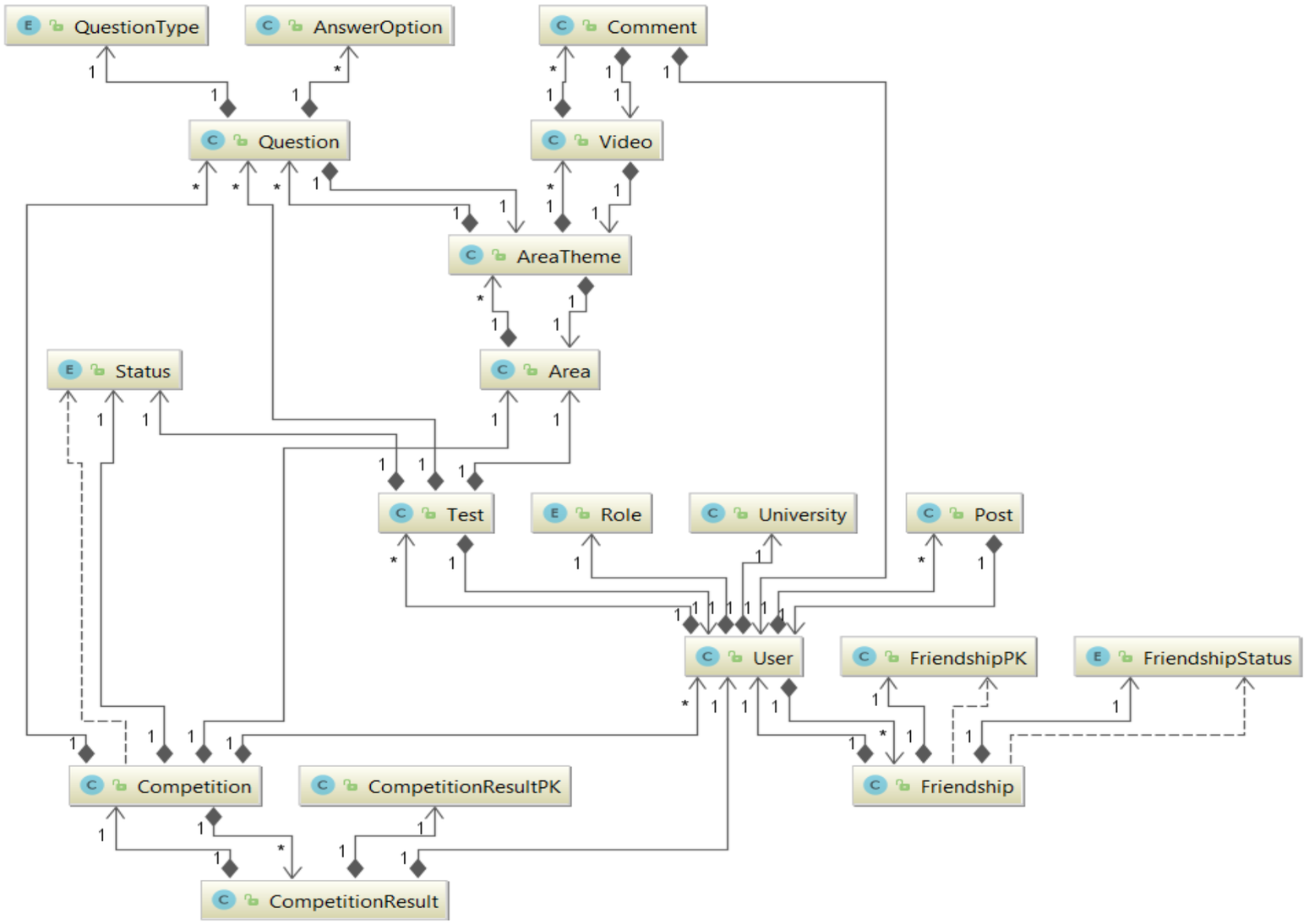
Система підготовки абітурієнтів до вступу

Схема зв’язків між класами

ІАЛЦ.467100.005 Д2

Аркушів 1

2020



					ІАЛЦ.467100.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	Схема зв’язків між класами	Літ. / Маса / Масштаб
Розроб.		Зварич Є.О.				
Перевір.		Габінет А.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П			Дипломна робота	КПІ ім.Ігоря Сікорського, ФІОТ кафедра ОТ гр. ІП-62
Затверд.						

ДОДАТОК З

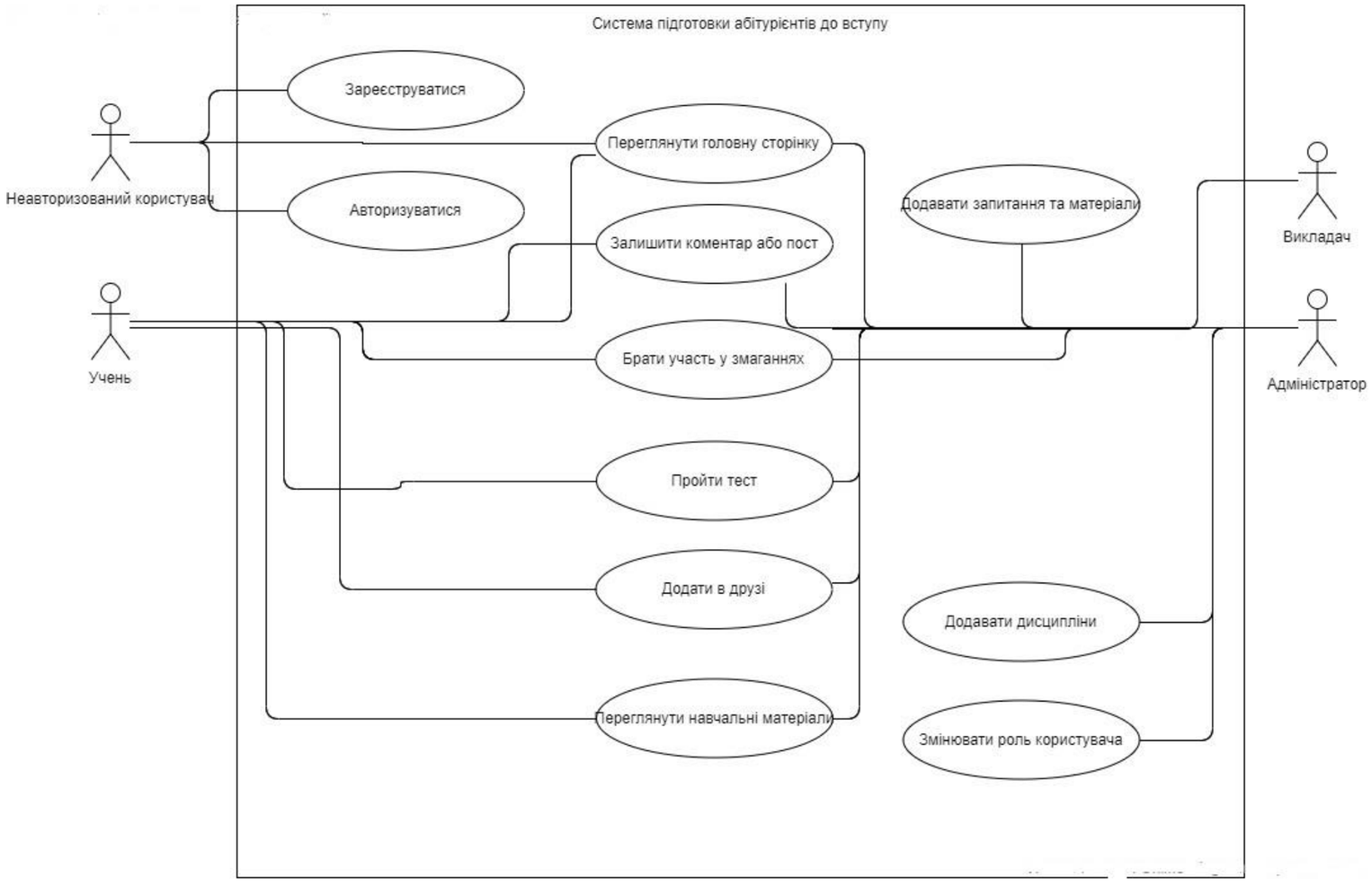
Система підготовки абітурієнтів до вступу

Схема прецедентів

ІАЛЦ.467100.006 ДЗ

Аркушів 1

2020



					ІАЛЦ.467100.006 ДЗ	
Зм.	Арк.	№ докум.	Підпис	Дата	Схема прецедентів	Літ. / Маса / Масштаб
Розроб.		Зварич Є.О.				
Перевір.		Габінет А.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П			Дипломна робота	КПІ ім.Ігоря Сікорського, ФІОТ кафедра ОТ гр. ІП-62
Затверд.						

ДОДАТОК 4

Система підготовки абітурієнтів до вступу

Текст програми

ІАЛЦ.467100.007 Д4

Аркушів 10

2020

Area.java

```
@Data
@Entity
@NoArgsConstructor
public class Area {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @Column(insertable = false, updatable = false, unique = true)
   private String title;

   private String imageUrl;

   @OneToMany(orphanRemoval = true, mappedBy = "area")
   private List<AreaTheme> areaThemes;

   public Area(Long id) {
      this.id = id;
   }
}
```

User.java

```
@Data
@NoArgsConstructor
@Entity
@Table(name = "app_user")
public class User {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @Column(unique = true, nullable = false)
   private String login;

   @Column(nullable = false)
   private String password;

   @Email
   @Column(unique = true)
   private String email;

   private Role role;

   private String realName;

   private Integer graduationYear;
```

```
    @ManyToOne
    private University desiredUniversity;

    private boolean enabled;

    @OneToMany(mappedBy = "user")
    private List<Test> tests;

    @OneToMany(mappedBy = "user")
    private List<Post> posts;

    @OneToMany(mappedBy = "invitedUser", cascade = CascadeType.ALL)
    private List<Friendship> invitedFriendships;

    @OneToMany(mappedBy = "acceptedUser", cascade = CascadeType.ALL)
    private List<Friendship> acceptedFriendships;

    public User(Long id) {
        this.id = id;
    }
}
```

Question.java

```
@Data
@Entity
@NoArgsConstructor
public class Question {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String text;

    @ManyToOne(optional = false)
    private AreaTheme areaTheme;

    @Column(name = "type_id")
    private QuestionType typeId;
    @Convert(converter = AnswerOptionConverter.class)
    @Column(columnDefinition = "text")
    private List<AnswerOption> answerOptions;

    public Question(Long id) {
        this.id = id;
    }
}
```

Test.java

```
@Data
@Entity
public class Test {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    private User user;

    @ManyToOne
    private Area questionArea;

    @Enumerated
    private Status status;

    private BigDecimal result;

    private Long startTime;

    private Long finishTime;

    @ManyToMany
    @JoinTable(name = "test_questions",
            joinColumns = { @JoinColumn(name = "question_id") },
            inverseJoinColumns = { @JoinColumn(name = "test_id") })
    private List<Question> testQuestions;

}
```

TestRepository.java

```
@Repository
public interface TestRepository extends JpaRepository<Test, Long> {

    @Query("select t from Test t " +
            "where t.status = com.testproject.test.domain.test.Status.FINISHED " +
            "order by t.result desc, (t.finishTime - t.startTime) asc")
    List<Test> getLeaderboandForArea(@Param("areaId") Long areaId, Pageable pageable);

    @Query("select t from Test t" +
            " where t.user.login = :userLogin" +
            " and t.questionArea.id = :areaId" +
            " and t.status = com.testproject.test.domain.test.Status.CONTINUE")
    Test getByUserAndAreaNonFinished(@Param("userLogin") String userLogin,
@Param("areaId") Long areaId);
```

```
    @Modifying
    @Query("delete from Test t where t.startTime > :time" +
            " and t.status = com.testproject.test.domain.test.Status.CONTINUE")
    void deleteOlderThan(@Param("time") long time);
}
```

TestServiceImpl.java

```
@Service
public class TestServiceImpl implements TestService {

    private static final int ONE_DAY_MILLI = 24 * 60 * 60 * 1000;

    @Autowired
    private QuestionsService questionsService;

    @Autowired
    private QuestionRepository questionRepository;

    @Autowired
    private TestRepository testRepository;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private CalculateFinalMarkService calculateFinalMarkService;

    @Autowired
    private TestMapper testMapper;

    @Autowired
    private QuestionMapper questionMapper;

    @Transactional
    @Override
    public NewTestOutputDto createNewTestForArea(Long areaId) {
        long currentTime = System.currentTimeMillis();

        long currentMinusDay = currentTime - ONE_DAY_MILLI;
        testRepository.deleteOlderThan(currentMinusDay);
        String login = SecurityContextHolder.getContext().getAuthentication().getName();
        User currentUser = userRepository.findByLogin(login).get();

        Test test = testRepository.getByUserAndAreaNonFinished(login, areaId);
        List<QuestionOutputDto> questionsForTest;
        if (test == null) {
            test = new Test();
            test.setQuestionArea(new Area(areaId));
```

```
            test.setStatus(Status.CONTINUE);
            test.setUser(currentUser);
            test.setStartTime(currentTime);

            questionsForTest = questionsService.getQuestionsForAreaTest(areaId);
            test.setTestQuestions(questionsForTest.stream()
                    .map(QuestionOutputDto::getId)
                    .map(Question::new).collect(Collectors.toList()));
            test = testRepository.save(test);
        } else {
            questionsForTest = test.getTestQuestions().stream()
                    .map(questionMapper::dtoFromQuestion)
                    .collect(Collectors.toList());
        }

        return new NewTestOutputDto(test.getId(), questionsForTest);
    }

    @Override
    public List<UserAnswerAnalysisDto> finishThemeTest(List<UserAnswerDto>
answerDtos) {
        List<Question> questionsList = questionRepository.findAllByIdIn(answerDtos.stream()
                .map(UserAnswerDto::getQuestionId).collect(Collectors.toSet()));
        Map<Long, List<String>> correctAnswersByQuestionId = questionsList.stream()
                .collect(Collectors.toMap(Question::getId, q ->

q.getAnswerOptions().stream().filter(AnswerOption::isCorrect).map(AnswerOption::getAnsw
er)
                        .collect(Collectors.toList())
                ));
        Map<Long, UserAnswerDto> userAnswersByQuestionId = answerDtos.stream()
                .collect(Collectors.toMap(UserAnswerDto::getQuestionId, Function.identity()));

        List<UserAnswerAnalysisDto> userAnswerAnalysisDtos = new ArrayList<>();

        for (Map.Entry<Long, List<String>> correctAnswer:
correctAnswersByQuestionId.entrySet()) {
            UserAnswerDto userAnswerDto = userAnswersByQuestionId
                    .getOrDefault(correctAnswer.getKey(), new UserAnswerDto());

            UserAnswerAnalysisDto analysisDto = new UserAnswerAnalysisDto();

            analysisDto.setUserAnswerDto(userAnswerDto);

analysisDto.setCorrect(correctAnswer.getValue().equals(userAnswerDto.getAnswers()));
            userAnswerAnalysisDtos.add(analysisDto);
        }

        return userAnswerAnalysisDtos;
```

```
    }

    @Override
    public TestResultAndAnalisysDto finishAreaTest(FinishTestInputDto finishTestInputDto)
{
        Test test =
testRepository.findById(finishTestInputDto.getTestId()).orElseThrow(IllegalArgumentExcept
ion::new);

        Map<Long, Question> questionMap = test.getTestQuestions().stream()
                .collect(Collectors.toMap(Question::getId, Function.identity()));
        Map<QuestionType, Long> questionsByType = test.getTestQuestions().stream()
                .collect(Collectors.groupingBy(Question::getTypeId, Collectors.counting()));
        Map<QuestionType, Long> correctAnswersByType = new HashMap<>();

        Map<Long, UserAnswerDto> userAnswersByQuestionId =
finishTestInputDto.getUserAnswerDtos().stream()
                .collect(Collectors.toMap(UserAnswerDto::getQuestionId, Function.identity()));

        List<UserAnswerAnalysisDto> userAnswerAnalysisDtos = new ArrayList<>();

        for (Map.Entry<Long, Question> questionEntry: questionMap.entrySet()) {
            Question q = questionEntry.getValue();
            List<String> answers = q.getAnswerOptions().stream()
                    .filter(AnswerOption::isCorrect)
                    .map(AnswerOption::getAnswer)
                    .collect(Collectors.toList());
            UserAnswerDto userAnswerDto = userAnswersByQuestionId
                    .getOrDefault(questionEntry.getKey(), new UserAnswerDto());

            UserAnswerAnalysisDto analysisDto = new UserAnswerAnalysisDto();
            analysisDto.setUserAnswerDto(userAnswerDto);

            boolean isCorrect = answers.equals(userAnswerDto.getAnswers());

            if (isCorrect) {
                if (correctAnswersByType.containsKey(questionEntry.getValue().getTypeId())) {
                    correctAnswersByType.put(q.getTypeId(),
correctAnswersByType.get(q.getTypeId()) + 1);
                } else {
                    correctAnswersByType.put(q.getTypeId(), 1L);
                }
                correctAnswersByType.get(questionEntry.getValue().getTypeId());
            }
            analysisDto.setCorrect(isCorrect);

            userAnswerAnalysisDtos.add(analysisDto);
        }
```

```
        BigDecimal mark = calculateFinalMarkService.execute(correctAnswersByType,
questionsByType);

        test.setStatus(Status.FINISHED);
        test.setFinishTime(System.currentTimeMillis());
        test.setResult(mark);
        testRepository.save(test);

        test.getQuestionArea();
        TestResultAndAnalisysDto testResultAndAnalisysDto = new
TestResultAndAnalisysDto();
        testResultAndAnalisysDto.setTestResult(testMapper.getResultFromTest(test));
        testResultAndAnalisysDto.setUserAnswerAnalysisList(userAnswerAnalysisDtos);

        return testResultAndAnalisysDto;
    }

    @Override
    public List<TestLeaderboardDto> getAreaLeaderboard(Long areaId) {
        return testRepository.getLeaderboandForArea(areaId, PageRequest.of(0, 10)).stream()
                .map(testMapper::toLeaderboard)
                .collect(Collectors.toList());
    }

}
```

CalculateFinalMarkService.java

```
@Component
public class CalculateFinalMarkServiceImpl implements CalculateFinalMarkService {
    @Override
    public BigDecimal execute(long numberOfCorrect, long allNumber) {
        return BigDecimal.valueOf(numberOfCorrect)
                .divide(BigDecimal.valueOf(allNumber), 2, RoundingMode.HALF_UP)
                .multiply(BigDecimal.valueOf(100));
    }

    @Override
    public BigDecimal execute(Map<QuestionType, Long> questionTypeToCorrect,
                              Map<QuestionType, Long> questionTypeToAll) {
        multValues(questionTypeToAll);
        multValues(questionTypeToCorrect);
        double correctPoints =
questionTypeToCorrect.entrySet().stream().mapToLong(Map.Entry::getValue).sum();
        double allPoints =
questionTypeToAll.entrySet().stream().mapToLong(Map.Entry::getValue).sum();
        return BigDecimal.valueOf(correctPoints / allPoints * 100).setScale(2,
RoundingMode.HALF_UP);
    }
```

```
private void multValues(Map<QuestionType, Long> questionTypeToValue) {
        questionTypeToValue.put(QuestionType.MULTI_ANSWER,
                questionTypeToValue.getOrDefault(QuestionType.MULTI_ANSWER, 0L) * 2);
        questionTypeToValue.put(QuestionType.OPEN,
                questionTypeToValue.getOrDefault(QuestionType.OPEN, 0L) * 3);
    }
}
```

TestController.java

```
@RestController
@RequestMapping("/tests")
public class TestController {

    @Autowired
    private QuestionsService questionsService;

    @Autowired
    private TestService testService;

    @GetMapping("/area/{areaId}")
    public ResponseEntity<NewTestOutputDto> startAreaTest(@PathVariable Long areaId) {
        return
ResponseEntity.status(HttpStatus.CREATED).body(testService.createNewTestForArea(areaId
));
    }

    @GetMapping("/theme/{themeId}")
    public List<QuestionOutputDto> getQuestionsForThemeTest(@PathVariable Long
themeId) {
        return questionsService.getQuestionsForThemeTest(themeId);
    }

    @PostMapping("/finish/theme")
    public List<UserAnswerAnalysisDto> finishThemeTest(@RequestBody
List<UserAnswerDto> answers) {
        return testService.finishThemeTest(answers);
    }

    @PostMapping("/finish/area")
    public TestResultAndAnalisysDto finishTest(@RequestBody FinishTestInputDto
testResults) {
        return testService.finishAreaTest(testResults);
    }

    @GetMapping("/leader/{areaId}")
    public List<TestLeaderboardDto> getLeaderboardForArea(@PathVariable Long areaId) {
        return testService.getAreaLeaderboard(areaId);
    }
}
```

TestService.ts

```
@Injectable({
  providedIn: 'root'
})
export class TestService {

  private readonly url = `${API_URL}/tests`;

  constructor(private http: HttpClient) {
  }

  createNewTestForArea(areaId: number): Observable<TestOutput> {
    return this.http.get<TestOutput>(`${this.url}/area/${areaId}`);
  }

  createNewTestForTheme(themeId: number): Observable<QuestionOutput[]> {
    return this.http.get<QuestionOutput[]>(`${this.url}/theme/${themeId}`);
  }

  getAreaThemeResult(test: number, userAnswers: UserAnswer[]):
Observable<TestResultAndAnalysis> {
    return this.http.post<TestResultAndAnalysis>(`${this.url}/finish/area`, {testId: test,
userAnswerDtos: userAnswers});
  }

  getThemeTestResult(userAnswers: UserAnswer[]):
Observable<UserAnswerAnalysisOutput[]> {
    return this.http.post<UserAnswerAnalysisOutput[]>(`${this.url}/finish/theme`,
userAnswers);
  }

  getLeaderboardForArea(areaId: number): Observable<LeaderboardOutput[]> {
    return this.http.get<LeaderboardOutput[]>(`${this.url}/leader/${areaId}`);
  }
}
```

show-area-test-component.ts

```
@Component({
  selector: 'app-show-area-test',
  templateUrl: './show-area-test.component.html',
  styleUrls: ['./show-area-test.component.css']
})
export class ShowAreaTestComponent implements OnInit {

  test: TestOutput = new TestOutput();

  userAnswers: UserAnswer[] = [];
```

```
  questionType = QuestionType;

  testIsFinished: Promise<boolean>;

  testResult: TestResultOutput = new TestResultOutput();

  analysis: any[] = [];
  analysisMap: Map<number, UserAnswerAnalysisOutput> = new Map<number,
UserAnswerAnalysisOutput>();

  constructor(private testService: TestService, private userAnswersHandlingService:
UserAnswersHandlingService,
              private route: ActivatedRoute) {
  }

  ngOnInit(): void {
    const id = +this.route.snapshot.paramMap.get('id');
    this.testService.createNewTestForArea(id).subscribe(t => {
      this.test = t;
      this.userAnswers = this.test.testQuestions.map(q => {
        const n = new UserAnswer();
        n.questionId = q.id;
        return n;
      });
    });
  }

  getResult() {
    this.userAnswers =
this.userAnswersHandlingService.mapAnswersOnQuestions(this.test.testQuestions,
this.userAnswers);
    this.testService.getAreaThemeResult(this.test.testId, this.userAnswers).subscribe((an:
TestResultAndAnalysis) => {
      this.testResult = an.testResult;
      this.analysis = an.userAnswerAnalysisList.map(a => new
UserAnswerAnalysisOutput(a.userAnswerDto, a.correct));
      this.analysis.forEach(value => this.analysisMap.set(value.userAnswerDto.questionId,
value));
      this.testIsFinished = Promise.resolve(true);
    });
  }

  setStyle() {
    return JSON.parse(localStorage.getItem('style'));
  }
```